SPECTRAL CONVOLUTIONAL NETWORKS ON HIERARCHICAL MULTIGRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spectral Graph Convolutional Networks (GCNs) are a generalization of convolutional networks to learning on graph-structured data. Applications of spectral GCNs have been successful, but *limited* to a few problems where the graph is fixed, such as shape correspondence and node classification. In this work, we address this limitation by revisiting a particular family of spectral graph networks, Chebyshev GCNs, showing its efficacy in solving graph classification tasks with a variable graph structure and size. Current GCNs also restrict graphs to have at most one edge between any pair of nodes. To this end, we propose a novel *multigraph* network that learns from multi-relational graphs. We explicitly model different types of edges: annotated edges, learned edges with abstract meaning, and hierarchical edges. We also experiment with different ways to fuse the representations extracted from different edge types. We achieve state-of-the-art results on a variety of chemical, social, and vision graph classification benchmarks.

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have seen wide success in domains where data is restricted to a Euclidean space. These methods exploit properties such as stationarity of the data distributions, locality and a well-defined notation of translation, but cannot model data that is non-Euclidean in nature. Such structure is naturally present in many domains, such as chemistry, social networks, transportation systems, and 3D geometry, and can be expressed by graphs (Bronstein et al., 2017; Hamilton et al., 2017b). By defining an operation on graphs analogous to convolution, Graph Convolutional Networks (GCNs) have extended CNNs to graph-based data. The earliest methods performed convolution in the spectral domain (Bruna et al., 2013), but subsequent work has proposed generalizations of convolution in the spatial domain. There have been multiple successful applications of GCNs to node classification (Velickovic et al., 2018) and link prediction (Schlichtkrull et al., 2018), whereas we target graph classification similarly to Simonovsky & Komodakis (2017).

Our focus is on multigraphs, a graph that is permitted to have multiple edges. Multigraphs are important in many domains, such as social networks. Some data, such as images, have inherent hierarchical structure, which we also represent as multigraphs, enabling us to exploit the hierarchy. The challenge of generalizing convolution to multigraphs is to have anisotropic convolution kernels (such as edge detectors). Anisotropic models, such as MoNet (Monti et al., 2017) and SplineCNN (Fey et al., 2018), rely on coordinate structure, work well for vision tasks, but are suboptimal for non-visual graph problems. Other general models exist (Battaglia et al., 2018), but making them efficient for a variety of tasks is impeded by the "no free lunch theorem" (Goodfellow et al., 2016, § 5.2.1).

Compared to non-spectral GCNs, spectral models have filters with more global support, which is important for capturing complex relationships. We rely on Chebyshev GCNs (ChebNet) (Defferrard et al., 2016). Even though it was originally derived from spectral methods Bruna et al. (2013), it does not suffer from their main shortcoming — sensitivity of learned filters to graph size and structure.

Contributions: We propose a scalable spectral GCN that learns from multigraphs by capturing multi-relational graph paths as well as multiplicative and additive interactions to reduce model complexity and learn richer representations. We also learn new abstract relationships between graph nodes, beyond the ones annotated in the datasets. To our knowledge, we are the first to demonstrate that spectral methods can efficiently solve problems with variable graph size and structure, where this kind of method is generally believed not to perform well.

2 MULTIGRAPH CONVOLUTION

While we provide the background to understand our model, a review of spectral graph methods is beyond the scope of this paper. Section 6.1 of the Appendix reviews spectral graph convolution.

2.1 APPROXIMATE SPECTRAL GRAPH CONVOLUTION

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of N nodes, $\mathcal{V}$, and edges, $\mathcal{E}$. Nodes $v_i \in \mathcal{V}$ usually represent specific semantic concepts such as atoms in a chemical compound or users in a social network. Nodes can also denote abstract blocks of information with common properties, such as superpixels in images. Edges $e_{ij} \in \mathcal{E}$ define the relationships and scope of which node effects propagate.

In spectral graph convolution (Bruna et al., 2013), the filter $g \in R^N$ is defined on an entire input space. Although it makes filters global, which helps to capture complex relationships, it is also desirable to have local support since the data often have local structure.

To address this issue, we can model this filter as a function of eigenvalues Λ (which is assumed to be constant) of the normalized symmetric graph Laplacian L: $g = g(\Lambda)$. We can further approximate it as a sum of K terms using the Chebyshev expansion, where each term $T_k(\Lambda) = 2\Lambda T_{k-1}(\Lambda) - T_{k-2}(\Lambda)$ contains powers Λ^k. Finally, we apply the property of eigendecomposition to eliminate computationally inconvenient eigenvectors $U \in \mathbb{R}^{N \times N}$ from spectral graph convolution:

$$L^k = (U\Lambda U^T)^k = U\Lambda^k U^T. \tag{1}$$

In general, for the input $X \in \mathbb{R}^{N \times X_{in}}$ with N nodes and X_{in}-dimensional features in each node, the approximate convolution is defined as:

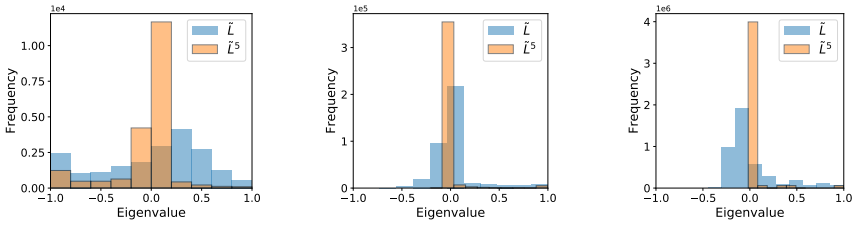
$$Y = \bar{X}\Theta, \tag{2}$$

where $\bar{X} \in \mathbb{R}^{N \times X_{in}K}$ are features projected onto the Chebyshev basis $T_k(\tilde{L})$ and concatenated for all orders $k \in [0, K-1]$ and $\Theta \in \mathbb{R}^{X_{in}K \times X_{out}}$ are trainable weights, where $\tilde{L} = L - I$.

This approximation scheme was proposed in Defferrard et al. (2016), and Eq. 2 defines the convolutional layer in the Chebyshev GCN (ChebNet), which is the basis for our method. Convolution is an essential computational block in graph networks, since it permits the gradual aggregation of information from neighboring nodes. By stacking the operator in Eq. 2, we capture increasingly larger neighborhoods and learn complex relationships in graph-structured data.

2.2 GRAPHS WITH VARIABLE STRUCTURE AND SIZE

The approximate spectral graph convolution (Eq. 2) enforces spatial locality of the filters by controlling the order of the Chebyshev polynomial K. Importantly, it reduces the computational complexity of spectral convolution from $O(N^2)$ to $O(K|\mathcal{E}|)$. In this work, we observe an important byproduct of this scheme: that learned filters become less sensitive to changes in graph structure and size due to excluding the eigenvectors U from spectral convolution, so that learned filters are not tied to U.



(a) ENZYMES (b) COLLAB (c) MNIST-75sp

Figure 1: Histograms of eigenvalues of the rescaled graph Laplacian $\tilde{L}$ for the (a) ENZYMES, (b) COLLAB and (c) MNIST (for 75 superpixels, see Section 3.2 for detail) datasets. Due to the property of eigendecomposition ($\tilde{L}^k = U\tilde{\Lambda}^k U^T$) the distribution of eigenvalues shrinks when we take powers of $\tilde{L}$ to compute the approximate spectral graph convolution (Eq. 2).

The only assumption that still makes a trainable filter $\hat{g}$ sensitive to graph structure is that we model it as a function of eigenvalues $\hat{g}(\Lambda)$. However, the distribution of eigenvalues of the normalized Laplacian is concentrated in a limited range, making it a weaker dependency on graphs than the spectral convolution via eigenvectors, so that learned filters generalize well to new graphs. Moreover, since we use powers of $\tilde{L}$ in performing convolution (Eq. 2), the distribution of eigenvalues $\tilde{\Lambda}$ further contracts due to exponentiation of the middle term on the RHS of Eq. 1. We believe that this effect accounts for the robustness of learned filters to changes in graph size or structure (Figure 1).

2.3 Graphs with multiple relation types

In the approximate spectral graph convolution (Eq. 2), the graph Laplacian $\tilde{L}$ encodes a single relation type between nodes. Yet, a graph may describe many types of distinct relations. In this section, we address this limitation by extending Eq. 2 to a multigraph, i.e. a graph with multiple ($R \geq 1$) edges (relations) between the same nodes encoded as a set of graph Laplacians $\{\tilde{L}^{(r)}\}_1^R$. Extensions to a multigraph can also be applied to early spectral models (Bruna et al., 2013) but, since ChebNet was shown to be superior in downstream tasks, we choose to focus on the latter model.

Two dimensional Chebyshev polynomial. The Chebyshev polynomial used in Eq. 2 (see Section 6.1 in Appendix for detail) can be extended for two variables (relations in our case) similarly to bilinear models, e.g., as in Omar et al. (2010):

$$T_{ij}(\tilde{L}^{(r_1)}, \tilde{L}^{(r_2)}) = T_i(\tilde{L}^{(r_1)})T_j(\tilde{L}^{(r_2)}), i, j = 0, ..., K-1, \tag{3}$$

and, analogously, for more variables. For $R = 2$, the convolution is then defined as:

$$Y = [\bar{X}_{0,0}, \bar{X}_{0,1}, ..., \bar{X}_{i,j}, ..., \bar{X}_{K-1,K-1}]\Theta, \tag{4}$$

where $\bar{X}_{i,j} = T_i(\tilde{L}^{(r_1)})T_j(\tilde{L}^{(r_2)})X$. In this case, we allow the model to leverage graph paths consisting of multiple relation types (Figure 2). This flexibility, however, comes at a great computational cost, which is prohibitive for a large number of relations R or large order K due to exponential growth of the number of parameters: $\Theta \in X_{in}K^R X_{out}$. Moreover, as we demonstrate in our experiments, such multi-relational paths do not necessary lead to better performance.

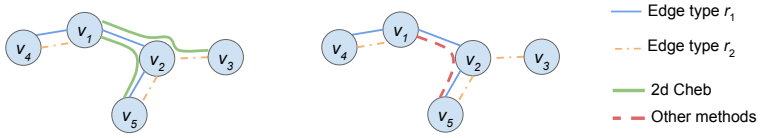
Multiplicative and additive fusion. Motivated by multimodal fusion considered in the Visual Question Answering literature (e.g. Kim et al., 2016), we propose the multiplicative operator:

$$Y = [f_0(\bar{X}^{(0)}) \circ f_1(\bar{X}^{(1)}) \circ ... \circ f_{R-1}(\bar{X}^{(R-1)})]\Theta. \tag{5}$$

In this case, node features interact in a multiplicative way. The advantage of this method is that the model can learn separate transformations f_r for each relation type r and has fewer trainable parameters preventing overfitting, which is especially important for large K and R. The element-wise multiplication $\circ$ in Eq. 5 can be replaced with summation to perform additive fusion.

Concatenating edge features. A more straightforward approach is to concatenate features $\bar{X}^{(r)}$ for all R relation types and learn a single matrix of weights $\Theta \in \mathbb{R}^{X_{in}KR \times X_{out}}$:

$$Y = [\bar{X}^{(0)}, \bar{X}^{(1)}, ..., \bar{X}^{(R-1)}]\Theta. \tag{6}$$



(a) (b)

Figure 2: Comparison of (a) the fusion method based on a two-dimensional Chebyshev polynomial (Eq. 3, 4) to (b) other proposed methods in case of a 2-hop filter (a filter averaging features of nodes located two edges away from the filter center - v_1 in this case). Note that (a) can leverage multi-relational paths and the filter centered at node v_1 can access features of the node v_3, which is not possible for other methods (b). In this work, edge type r_1 can denote annotated or spatial relations, while r_2 denotes hierarchical or learned ones. We also allow for three and more relation types.

This method, however, does not scale well for large R, since the dimensionality of Θ grows linearly with R. Note that even though multi-relational paths are not explicit in Eq. 5 and 6, for a *multi-layer* network, relation types will still communicate through node features. In Figure 2, node v_2 will contain features of node v_3 after the first convolutional layer, so that in the second layer the filter centered at node v_1 will have access to features of node v_3 by accessing features of node v_2. Compared to the 2d polynomial convolution defined by Eq. 4, the concatenation-based, multiplicative and additive approaches require more layers to have a larger multi-relational receptive field.

3 MULTIGRAPH CONVOLUTIONAL NETWORKS

A popular 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, for example, whether two atoms in a molecule are bonded (Duvenaud et al., 2015). 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 annotated in the data.

3.1 LEARNING EDGES

Prior work (e.g. Schlichtkrull et al., 2018; Bordes et al., 2013), proposed methods to learn from multiple edges, but similarly to the methods using a single edge type (Kipf & Welling, 2016), they leveraged only predefined (annotated) edges in the data. We devise a more flexible model, which, in addition to learning from an arbitrary number of predefined relations between nodes (see Section 2.3), learns abstract edges jointly with a GCN. We propose to learn a new edge $e_{ij}^{(r)}$ between any pair of nodes v_i and v_j with features X_i and X_j using a trainable similarity function:

$$e_{ij}^{(r)} = \frac{\exp\left(f_{\text{edge}}\left(X_i, X_j\right)\right)}{\sum_{k \in [1, |\mathcal{V}|]} \exp\left(f_{\text{edge}}\left(X_i, X_k\right)\right)}, \tag{7}$$

where the softmax is used to enforce sparse connections. This idea is similar to Henaff et al. (2015), built on the early spectral convolution model (Bruna et al., 2013), which learned an adjacency matrix, but targeted classification tasks for non graph-structured data (e.g. document classification, with each document represented as a feature vector). Moreover, we learn this matrix jointly with a more recent graph classification model (Defferrard et al., 2016) and, additionally, efficiently fuse predefined and learned relations. Eq. 7 is also similar to that of Velickovic et al. (2018), which used this functional form to predict an attention coefficient α_{ij} for some *existing* edge e_{ij}. The attention model can only strengthen or weaken some existing relations, but cannot form new relations. We present a more general model that makes it possible to connect previously disconnected nodes and form *new* abstract relations. To enforce a symmetry of predicted edges we compute an average: $(e_{ij}^{(r)} + e_{ji}^{(r)})/2$.

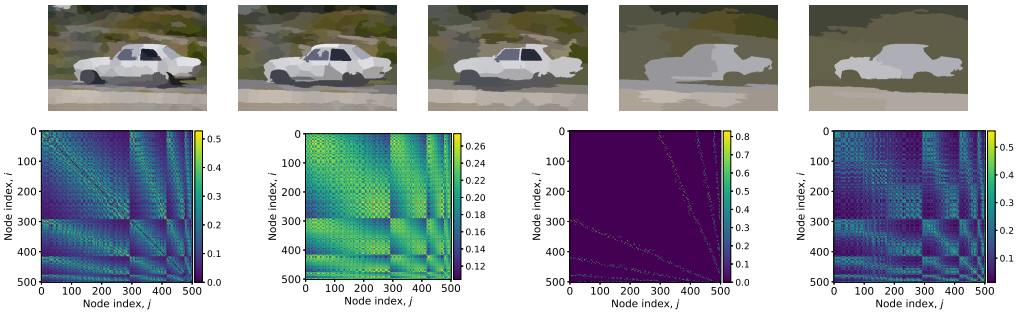
3.2 HIERARCHICAL EDGES FOR IMAGES

Image classification was recently formulated as a graph classification problem in Defferrard et al. (2016); Monti et al. (2017); Fey et al. (2018), who considered small-scale image classification problems such as MNIST (LeCun et al., 1998). In this work, we present a model that scales to more complex and larger image datasets, such as PASCAL VOC 2012 (Everingham et al., 2010). We follow Monti et al. (2017) and compute SLIC (Achanta et al., 2012) superpixels for each image and build a graph, in which each node v_i corresponds to a superpixel and edges e_{ij} are computed based on the Euclidean distance between the coordinates p_i and p_j of their centers of masses:

$$e_{ij} = \exp(-\frac{||p_i - p_j||^2}{2\sigma^2}). \tag{8}$$

Next, we introduce a novel hierarchical graph model (Figure 3). We compute superpixels at several different scales and create children-parent relations based on intersection over union (IoU) between superpixels v_i and v_j at different scales encoded as an adjacency matrix $A^{(r_{\text{hier}})}$:

$$A_{ij}^{(r_{\text{hier}})} = IoU(v_i, v_j), \tag{9}$$



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

Figure 3: *Top row*: We compute superpixels at several scales and combine all of them into a single set. We then build a graph, where each node corresponds to a superpixel from this set and has associated features, such as mean RGB color, coordinates of center of masses or convolutional VGG-16 features. *Bottom row*: Using Eq. 8 and 9, we compute spatial (a) and hierarchical (c) edges between nodes represented as scaled graph Laplacians $\tilde{L}$. Nodes 0 to 300 correspond to the first level of hierarchy (first scale of superpixels), and 300 to 400 corresponds to the second level, and so forth. Notice that spatial edges (a) are created both within and across the levels, while hierarchical (c) edges exist only between hierarchical levels. (c, d) Powers of these graph Laplacians $\tilde{L}^{K=7}$ allow edges to diffuse over the graph making it possible to learn filters with more global support.

where $A_{ij}^{(r_{\text{hier}})} = 0$ for superpixels at the same scale. Spatial relations are computed as in flat models using Eq. 8, treating superpixels at different scales as a joint set of superpixels.

3.3 Layer pooling versus global pooling

Inspired by convolutional networks, previous works (Bruna et al., 2013; Defferrard et al., 2016; Monti et al., 2017; Simonovsky & Komodakis, 2017; Fey et al., 2018) built an analogy of pooling layers in graphs, for example, using the Graclus clustering algorithm (Dhillon et al., 2007). 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 in a downstream task (Springenberg et al., 2014) — 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 Dhillon et al. (2007), is also computationally intensive since we need to run the clustering algorithm for each training example independently, which limits the scale of problems we can address. Aiming to simplify the model while maintaining classification accuracy, we exclude pooling layers between conv. layers and perform global maximum pooling (GMP) over nodes following the last conv. layer. This fixes the size of the penultimate feature vector regardless of the number of nodes (Figure 4).

4 Experiments

We evaluate our model on chemical, social, and image-based graph classification datasets. For each dataset, there is a set of graphs with an arbitrary number of nodes $N = |\mathcal{V}|$ and undirected edges $|\mathcal{E}|$ of a single type ($R = 1$) and each graph $\mathcal{G}$ has a single, categorical label that is to be predicted. Dataset statistics are presented in Table 3 of the Appendix.

4.1 Dataset details

Chemical compounds. We consider five chemical compound datasets frequently used in previous work: NCI1 and NCI109 (Wale et al., 2008), MUTAG (Debnath et al., 1991), ENZYMES (Schomburg et al., 2004), and PROTEINS (Borgwardt et al., 2005). Every graph represents some chemical compound labeled according to its functional properties.

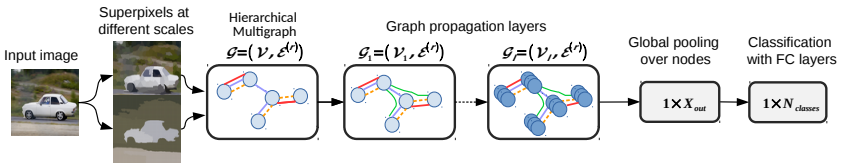


Figure 4: Graph classification pipeline for images. Each l^{th} convolutional layer in our model takes the graph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}^{(r)})$ and returns a graph with the same nodes and edges. Node features become increasingly global after each subsequent layer as the receptive 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 neighbors and the entire graph. By pooling over nodes we summarize the information collected by each node. Fully-connected layers follow global pooling to perform classification. For chemical/social datasets we use a similar pipeline, but the input graph $\mathcal{G}$ is provided by the dataset rather than built from images and does not have hierarchical edges.

Social networks. We experiment with three social network datasets. COLLAB (Leskovec et al., 2007; Shrivastava & Li, 2014) is a scientific collaboration dataset of researchers publishing in different subfields of physics. IMDB-B (Yanardag & Vishwanathan, 2015) is a movie collaboration dataset of actors/actresses appearing in different genres of movies. IMDB-M is similar to IMDB-B, but has more graphs and more genres. Each graph in the social datasets corresponds to an ego-network of a researcher or actor/actress and the task is to predict the subfield of physics for COLLAB or a genre of movies for IMDB. Graph nodes are featureless, i.e. each example (ego-network) in the dataset is defined only by an adjacency matrix A. To add features to some i^{th} node, we use the square root of node degree $\sqrt{D_i}$, which was shown to be a strong feature in Simonovsky & Komodakis (2017), despite its simplicity.

The chemical and social datasets vary in the number of graphs (188 - 5000) and class labels (2 - 6) and, thereby, represent a comprehensive benchmark for our method. We follow the standard approach to evaluation (Shervashidze et al., 2011; Yanardag & Vishwanathan, 2015) and perform 10-fold cross-validation on these datasets reporting average classification accuracies.

Images. To demonstrate the scalability of our model and its ability to leverage hierarchical edges, we experiment with three image datasets. MNIST (LeCun et al., 1998) consists of 70k grayscale 28×28 px images of handwritten digits. CIFAR-10 (Krizhevsky, 2009) has 60k colored 32×32 px images of 10 categories of animals and vehicles. PASCAL Visual Object Classes 2012 (Everingham et al., 2010) is a more challenging dataset with realistic high resolution images (up to 500×500 px) 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 4.3).

4.2 Architectural details and experimental setup

In all experiments, we train a ChebNet with three graph convolutional layers followed by global max pooling (GMP) and 1-2 fully-connected layers (Figure 4). Batch normalization (BN) and the ReLU activation are added after each layer, whereas dropout is added only before the fully-connected layers. Projections $f_r(\bar{X}^{(r)})$ in Eq. 5 are modeled by a single layer neural network with $C = 128$ hidden units and the $\tanh$ activation. The edge prediction function f_{edge} (see Eq. 7, Section 3.1) is a two layer neural network with 128 hidden units, which acts on concatenated node features. Detailed network architectures are presented in Table 3 of the Appendix.

We train all models using the Adam optimizer (Kingma & Ba, 2014) with learning rate of 0.001, weight decay of 0.0001, and batch size of 32. For chemical and social datasets, the learning rate is decayed after 25, 35, and 45 epochs and the models are trained for 50 epochs as in Simonovsky & Komodakis (2017). For visual datasets, the learning rate is decayed after 17 epochs and the models are trained for a total of 25 epochs, similar to Monti et al. (2017).

Table 1: Chemical (left five columns) and social (right three columns) graph classification results (accuracy in %). Multigraph ChebNet obtains better results by leveraging two types of edges: annotated and learned, whereas all other models use only annotated edges. *We implemented MoNet, GCN and ChebNet. To make a fair comparison to Multigraph ChebNet, we use the same network architectures, batch-normalization, global max pooling and node degree features (for the social datasets). For MoNet, coordinates are defined using node degrees as in Monti et al. (2017). The top result across all methods for each dataset is underlined and bolded, second from the top is bolded.

Model	NCI1	NCI109	MUTAG	ENZYMES	PROTEINS	COLLAB	IMDB-B	IMDB-M
WL[1]	**84.6**	**84.5**	83.8	59.1	–	–	–	–
structure2vec[2]	83.7	**82.2**	88.3	**61.1**	–	–	–	–
DGK[3]	80.3	80.3	87.4	53.4	75.7	73.1	67.0	44.6
PSCN[4]	78.6	–	**92.6**	–	75.9	72.6	71.0	45.2
DGCNN[5]	74.4	–	85.8	–	**76.3**	73.8	70.0	47.8
MoNet[6] - ours*	69.9	69.9	86.1	38.3	71.2	63.7	69.8	44.9
GCN[7]- ours*	75.9	72.8	77.7	39.7	74.4	70.2	**72.5**	47.9
ChebNet[8] - ours*	83.1	82.0	85.7	58.7	75.7	**76.0**	72.3	**48.4**
Multigraph ChebNet	**84.0**	**82.2**	**91.5**	**64.7**	**76.4**	**78.6**	**73.7**	**49.7**

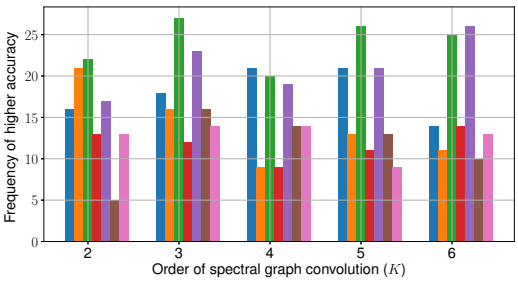
[1] Shervashidze et al. (2011); [2] Dai et al. (2016); [3] Yanardag & Vishwanathan (2015); [4] Niepert et al. (2016); [5] Zhang et al. (2018); [6] Monti et al. (2017); [7] Kipf & Welling (2016); [8] Defferrard et al. (2016)

For chemical and social datasets, we run cross-validation for different fusion methods (Section 2.3) and Chebyshev orders K in range from 2 to 6 (Section 2.1), while for image datasets, we train the models for $K = 7$ for superpixels in case of MNIST and CIFAR-10 and for $K = 15, 20, 25$ (respectively for each convolutional layer) for full grid MNIST.

Graph formation for images. For MNIST, we compute a hierarchy of 75, 21, and 7 SLIC superpixels and use mean superpixel intensity as node features, $X \in \mathbb{R}^{N\times 1}$. For CIFAR-10, we add two more levels of 300 and 150 superpixels and add coordinates features, $X \in \mathbb{R}^{N\times 5}$, because images are more complex. For PASCAL, due to its even more challenging images, we further add one more level of 1,000 superpixels and add features from the last convolutional layer of VGG-16 (Simonyan & Zisserman, 2014) pre-trained on ImageNet (Russakovsky et al., 2015). We feed images of original resolution to VGG-16 and add a bilinear upsampling layer on top of the last convolutional layer to obtain feature maps of the same spatial size as the input images. We then use superpixel masks to extract features for each superpixel and average features inside each mask, so that we have a fixed 512 dimensional vector for each superpixel. These are concatenated with the superpixel coordinates and fed to the graph convolutional model, i.e. $X \in \mathbb{R}^{N\times 515}$. VGG-16 features are very descriptive and local features already contain global image features, which are usually captured with higher orders of K (more global filters) in our model, so we found $K = 2$ to be optimal.

4.3 Results

Chemical and social graph classification: on five chemical compounds datasets, previous works typically show strong performance on one or two datasets (Table 1). In contrast, the Multigraph ChebNet, leveraging two relation types (annotated and learned, see Section 3.1), shows high accuracy across all datasets. On two chemical datasets, ENZYMES and PROTEINS, we outperform all previous methods. We also obtain competitive accuracy on NCI1 outperforming all but one prior work (Shervashidze et al., 2011). Importantly, the Multigraph ChebNet with two edge types, i.e. predefined dataset annotations and the learned edges (Section 3.1) consistently outperforms the baseline ChebNet with a single edge, which shows efficacy of our approach and demonstrates the complementary nature of predefined and learned edges. Lower results on NCI1 and NCI109 can be explained by the fact that the node features in the graphs of these datasets are imbalanced with some features appearing only a few times in the dataset. This is undesirable for our method, which learns new edges based on features and the model can predict random values for unseen features. On MUTAG we surpass all but one method Niepert et al. (2016). But in this case the dataset is tiny, consisting of 188 graphs and the margin from the top method is not statistically significant. On the three social datasets, COLLAB, IMDB-B and IMDB-M, we win by a large margin compared to previous works (Table 1). This high performance can be partially explained by using degree features, but we still observe a further boost by using learned edges in a Multigraph ChebNet.



Fusion method	# of parameters
Single edge* (Eq. 2)	$X_{in}KX_{out}$
Concat (Eq. 6)	$X_{in}KRX_{out}$
2d Cheb (Eq. 4)	$X_{in}K^{R}X_{out}$
Multiply (Eq. 5)	$C(X_{in}KR + X_{out})$
Sum (Eq. 5)	$C(X_{in}KR + X_{out})$
M-shared (Eq. 5)	$C(X_{in}K + X_{out})$
S-shared (Eq. 5)	$C(X_{in}K + X_{out})$

Figure 5: Comparison of edge fusion methods on chemical and social datasets for 10 folds. We observe that some methods perform well for lower order K, such as 2d Chebyshev convolution winning in 21/80 cases for $K = 2$, while others perform better for higher K, such as `Multiply-shared`. `Multiply` generally performs well across different K. All fusion methods, except for `Sum-shared`, outperform the `Single-edge` baseline. We also show the number of parameters in Θ for the Chebyshev convolution layer depending on the number of input features X_{in}, number of output features X_{out}, number of relation types R, order K and some constant C (number of hidden units in a projection layer f_r in Eq. 5). *`Single-edge` denotes using only annotated edges. All other methods additionally use the second edge, learned based on node features.

Evaluation of edge fusion methods. For the chemical and social datasets, we train a model using each of the edge fusion methods proposed in Section 2.3 and report the summary of results in Figure 5. We count the number of times each method outperforms the others treating all 10 folds independently. As expected, graph convolution based on the two-dimensional Chebyshev polynomial is better for lower orders of K, since it exploits multi-relational graph paths, effectively increasing the receptive field of filters. However, for larger K, the model complexity becomes too high due to quadratic growth of the number of parameters and performance degrades. Sharing weights for multiplicative or additive fusion generally drops performance with a few exceptions for social datasets in the multiplicative case. This implies that predefined and learned edges are of a different nature. It would be interesting to validate these fusion methods on a larger number of relation types.

Image classification. For image classification tasks, we observe several interesting results (Table 2). First, by adding batch-normalization and global max pooling, and increasing the learning rate to 0.001 compared to 0.0001 in Monti et al. (2017), accuracies surge both for MoNet (Monti et al., 2017) and ChebNet (Defferrard et al., 2016) (to 98.59% and 97.12% respectively from the reported 91.11% and 75.62%) reducing the gap between the two methods (from 15.49% to 1.47%). It supports our claim that spectral methods, in particular ChebNet, can efficiently perform graph classification for arbitrary sized graphs. Here, the arbitrariness is due to the number of superpixels being the upper bound in all cases, since SLIC often returns fewer superpixels.

Next, we observe that the hierarchical connections modeled by Hierarchical ChebNet are important for MNIST and CIFAR-10, substantially improving accuracy. Learned connections implemented by Multigraph ChebNet also provide a boost in performance, but in a less consistent way. For MNIST, they contribute almost 1% in the case of full grid, while only $\sim$0.3% in the case of superpixels. Combining hierarchical connections and multiple edge types (Hierarchical Multigraph ChebNet), we observe a further gain in performance, however, it is less pronounced and less consistent.

For PASCAL VOC, we obtain the best performance using Multigraph ChebNet with $R = 2$ relation types. Interestingly, our results are better in this case compared to the baseline ConvNet. The baseline results were obtained by stacking and training three convolutional layers (with the same number of filters as in ChebNet) on top of the last convolutional layer in VGG-16 to make a more fair comparison to our graph network, so that both the ConvNet and ChebNet take the same input. The ChebNet, however, uses coordinates as additional features, which can explain the phenomenon.

5 Related work and Discussion

Our method relies on a fast approximate spectral graph convolution known as ChebNet (Defferrard et al., 2016)), which was designed for graph classification. A simplified and faster version of this

Table 2: Image classification results. *Compared to tne implementation of ChebNet in Defferrard et al. (2016); Monti et al. (2017), our implementation has GMP without any pooling between convolutional layers, BN, a different learning rate and filter order K. R is the number of relation types. † LeCun et al. (1998) for MNIST, Hinton et al. (2012) for CIFAR-10, VGG-16 (Simonyan & Zisserman, 2014) pre-trained on ImageNet for PASCAL. ‡On 3× GTX Titan Black GPUs in PyTorch.

Model	R	**MNIST**		**CIFAR-10**		**PASCAL**
Node features		mean color		mean color + coord		vgg16 + coord
Graph		Full grid	75 sp	Full grid	300 sp	1000 sp
ConvNet†	–	99.33	–	84.40	–	85.88
GCN (Kipf & Welling, 2016)-ours*	1	79.28	80.35	43.94	45.51	84.25
MoNet (Monti et al., 2017)-ours*	1	$>$ 7 days‡	**98.59**	$>$ 7 days‡	**75.23**	80.05
ChebNet (Defferrard et al., 2016)-ours*	1	98.45	97.12	72.15	69.95	86.41
Multigraph ChebNet	2	**99.36**	97.43	72.98	70.96	**86.78**
Hierarchical ChebNet	2	99.09	97.79	**74.39**	73.15	**86.55**
Hierarchical Multigraph ChebNet	3	**99.24**	**98.16**	**75.67**	**73.35**	86.43

model, Graph Convolutional Networks (GCN) Kipf & Welling (2016), which is practically equivalent to the ChebNet with order $K = 1$, has shown impressive node classification performance on citation and knowledge graph datasets in the transductive learning setting. In all our experiments, we noticed that using more global filters (with larger K) is important (Tables 1, 2). Other recent works Hamilton et al. (2017a); Velickovic et al. (2018) also focus on node classification and, therefore, are not empirically compared to in this work.

Closely related to our work, Monti et al. (2017) formulated the generalized graph convolution model (MoNet) based on a trainable transformation to pseudo-coordinates, which led to learning anisotropic kernels and excellent results in visual tasks. In some cases, it outperforms our model in image classification (Table 2). However, in non-visual tasks, when coordinates are not naturally defined, the performance gain is less pronounced (Table 1). Notably, the computational cost (both memory and speed) of MoNet is higher than for ChebNet due to the patch operator in Monti et al. (2017, Eq. (9)-(11)). The argument in favor of MoNet against ChebNet was the sensitivity of spectral convolution methods, including ChebNet, to changes in graph size and structure. We contradict this argument and show comparable performance on visual tasks and better performance on chemical and social graph classification datasets. SplineCNN (Fey et al., 2018) is similar to MoNet and is good at classifying both graphs and nodes, but it is also based on pseudo coordinates and, therefore, has the same shortcoming of MoNet. So, its performance on general graph classification problems where coordinates are not well defined is expected to be inferior.

6 Conclusion

In this work, we address several limitations of current graph convolutional networks and show superior graph classification results on a number of chemical, social, and image-based datasets. First, we revisit the spectral graph convolution model based on the Chebyshev polynomial, commonly believed to inherit shortcomings of earlier spectral methods, and demonstrate its ability to learn from graphs of arbitrary size and structure. Second, we design and study edge fusion methods for multi-relational graphs, and show the importance of validating these methods for each task to achieve optimal performance. Third, we propose a way to learn new edges in a graph jointly with a graph classification model. Our results show that the learned edges are complimentary to edges already annotated, providing a significant gain in accuracy. Finally, we show that by adding hierarchical edges for images and fusing them with spatial and learned edges, we can further improve results.

REFERENCES

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. 2012.

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.

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*, pp. 2787–2795, 2013.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

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.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pp. 2702–2711, 2016.

Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

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.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

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.

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*, pp. 869–877, 2018.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017a.

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

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

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

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. URL `http://graphkernels.cs.tu-dortmund.de`.

Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

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.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

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, pp. 3, 2017.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023, 2016.

Zaid Omar, Nikolaos Mitianoudis, and Tania Stathaki. Two-dimensional chebyshev polynomials for image fusion. 2010.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

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*, pp. 593–607. Springer, 2018.

Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004.

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.

Anshumali Shrivastava and Ping Li. A new space for comparing graphs. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 62–71. IEEE Press, 2014.

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. 2018.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

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

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI Conference on Artificial Inteligence*, 2018.

APPENDIX

6.1 OVERVIEW OF SPECTRAL GRAPH CONVOLUTION AND ITS APPROXIMATION

Following the notation of Defferrard et al. (2016), spectral convolution on a graph $\mathcal{G}$ having N nodes is defined analogously to convolution in the Fourier domain (the convolution theorem) for some one-dimensional features over nodes $x \in \mathbb{R}^N$ and filter $g \in \mathbb{R}^N$ as (Bruna et al., 2013; Bronstein et al., 2017):

$$y = g \star x = U(U^T g \circ U^T x) = U\text{diag}(\hat{g})U^T x, \tag{10}$$

where, U are the eigenvectors of the normalized symmetric graph Laplacian, $L = I - D^{-1/2}AD^{-1/2}$, where A is an adjacency matrix of the graph $\mathcal{G}$, D are node degrees. $L = U\Lambda U^T$ follows from the definition of eigenvectors, where Λ is a diagonal matrix of eigenvalues. The operator $\circ$ denotes the Hadamard product (element-wise multiplication), $\hat{g} = U^T g$ and $\text{diag}(\hat{g})$ is a diagonal matrix with elements of $\hat{g}$ in the diagonal.

The spectral convolution in (10) can be approximated using the Chebyshev expansion, where $T_k(\Lambda) = 2\Lambda T_{k-1}(\Lambda) - T_{k-2}(\Lambda)$ with $T_0(\Lambda) = 1$ and $T_1(\Lambda) = \Lambda$ (i.e. $T_k(\Lambda)$ terms contain powers Λ^k) and the property of eigendecomposition:

$$L^k = (U\Lambda U^T)^k = U\Lambda^k U^T. \tag{11}$$

Assuming eigenvalues Λ are fixed constants, filter $\hat{g}$ can be represented as a function of eigenvalues $\hat{g}(\Lambda)$, such that (10) becomes:

$$g \star x = U\hat{g}(\Lambda)U^T x. \tag{12}$$

Filter $\hat{g}(\Lambda)$ can be further approximated as a Chebyshev polynomial of degree K (a weighted sum of $T_k(\Lambda)$ terms). Substituting the approximated $\hat{g}(\Lambda)$ into Eq. 12 and exploiting Eq. 11, the approximate spectral convolution takes the form of (see in Defferrard et al. (2016); Kipf & Welling (2016) for further analysis and Hammond et al. (2011) for derivations):

$$y = g \star x \approx U \left[\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \right] U^T x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) x = [\bar{x}_0, \bar{x}_1, ..., \bar{x}_{K-1}]\theta, \tag{13}$$

where $\tilde{L} = 2L/\lambda_{\max} - I$ is a rescaled graph Laplacian with $\lambda_{\max}$ as the largest eigenvalue of L, $\bar{x}_k = T_k(\tilde{L})x \in \mathbb{R}^N$ are projections of input features onto the Chebyshev basis and $\theta = [\theta_0, \theta_1, ..., \theta_{K-1}]$ are learnable weights shared across nodes. In this work, we further simplify the computation and fix $\lambda_{\max} = 2$ ($\lambda_{\max}$ varies from graph to graph), so that $\tilde{L} = L - I = -D^{-1/2}AD^{-1/2}$ and assume no loops in a graph. $\tilde{L}$ has the same eigenvectors U as L, but its eigenvalues are $\tilde{\Lambda} = \Lambda - 1$.

6.2 DATASET STATISTICS AND NETWORK ARCHITECTURES

Table 3: Dataset statistics and graph network architectures. For chemical and social datasets these statistics can also be found in Kersting et al. (2016) along with the datasets themselves. N - number of nodes in a graph. GC - graph convolution layer, FC - fully connected layer, D - dropout.

Dataset	**# graphs**	N_{min}	N_{max}	N_{avg}	**Architecture**
NCI1	4110	3	111	29.87	GC32-GC64-GC128-D0.1-FC256-D0.1-FC2
NCI109	4127	4	111	29.68	GC32-GC64-GC128-D0.1-FC256-D0.1-FC2
MUTAG	188	10	28	17.93	GC32-GC32-GC32-D0.1-FC96-D0.1-FC2
ENZYMES	600	2	126	32.63	GC32-GC64-GC512-D0.1-FC256-D0.1-FC6
PROTEINS	1113	4	620	39.06	GC32-GC32-GC32-D0.2-FC48-D0.2-FC2
COLLAB	5000	32	49	74.49	GC32-GC32-GC32-D0.2-FC48-D0.2-FC3
IMDB-B	1000	12	136	19.77	GC32-GC32-GC32-D0.2-FC48-D0.2-FC2
IMDB-M	1500	7	89	13.00	GC32-GC32-GC32-D0.1-FC64-D0.1-FC3
MNIST	70,000	40	75	70.57	GC32-GC64-GC512-D0.5-FC10
CIFAR-10	60,000	192	300	272.33	GC32-GC64-GC512-D0.5-FC10
PASCAL	11,540	57	1000	822.09	GC32-GC64-GC512-D0.7-FC20