Differentiable Graph Module (DGM) for Graph Convolutional Networks

Anees Kazi [1]   Luca Cosmo [2]   Nassir Navab [1 3]   Michael Bronstein [4 2 5]

Abstract

Graph deep learning has recently emerged as a powerful ML concept allowing to generalize successful deep neural architectures to non-Euclidean structured data. Such methods have shown promising results on a broad spectrum of applications ranging from social science, biomedicine, and particle physics to computer vision, graphics, and chemistry. One of the limitations of the majority of current graph neural network architectures is that they are often restricted to the transductive setting and rely on the assumption that the underlying graph is *known* and *fixed*. In many settings, such as those arising in medical and healthcare applications, this assumption is not necessarily true since the graph may be noisy, partially- or even completely unknown, and one is thus interested in inferring it from the data. This is especially important in inductive settings when dealing with nodes not present in the graph at training time. Furthermore, sometimes such a graph itself may convey insights that are even more important than the downstream task. In this paper, we introduce Differentiable Graph Module (DGM), a learnable function predicting the edge probability in the graph relevant for the task, that can be combined with convolutional graph neural network layers and trained in an end-to-end fashion. We provide an extensive evaluation of applications from the domains of healthcare (disease prediction), brain imaging (gender and age prediction), computer graphics (3D point cloud segmentation), and computer vision (zero-shot learning). We show that our model provides a significant improvement over baselines both in transductive and inductive settings and achieves state-of-the-art results.

[1]Computer Aided Medical Procedures (CAMP), Technical University of Munich, Germany [2]Imperial College London [3]Whiting School of Engineering, Johns Hopkins University, Baltimore, USA [4]University of Lugano, Switzerland [5]UK 4Twitter,UK. Correspondence to: Anees Kazi <anees.kazi@tum.de>.

None

1. Introduction

Geometric deep learning (GDL) is a novel emerging branch of deep learning attempting to generalize deep neural networks to non-Euclidean structured data such as graphs and manifolds (Bronstein et al., 2017; Hamilton et al., 2017b; Battaglia et al., 2018). Graphs in particular, being a very general abstract descriptions of relation and interaction systems, are ubiquitous in different branches of science. Graph-based learning models have been successfully applied in social networks, link prediction (Zhang & Chen, 2018), human-object interaction (Qi et al., 2018), computer vision (Qi et al., 2017) and graphics (Monti et al., 2017; Wang et al., 2019), particle physics (Choma et al., 2018), chemistry (Duvenaud et al., 2015; Gilmer et al.; Li et al., 2018b), medicine (Parisot et al., 2018; 2017; Mellema et al., 2019; Kazi et al., 2019b), drug repositioning (Zitnik et al., 2018), and protein science (Gainza et al., 2019), to mention a few.

Early formulations of learning on graphs date back to the seminal work of (Scarselli et al., 2008). Bruna et al. (2013) proposed formulating convolution-like operations in the spectral domain, defined by the eigenvectors of the graph Laplacian operator, a non-Euclidean analogy of the Fourier transform. More efficient and generalizable spectral graph CNNs were developed using polynomial (Defferrard et al., 2016; Kipf & Welling, 2016) or rational (Levie et al., 2018; Bianchi et al., 2019) spectral filters. Monti et al. (2017) proposed an analogy of 'patches' on graphs based on local weighting. A similar model was employed in graph attention networks (Veličković et al., 2017). The graph attention mechanism was extended in follow-up papers (Kondor, 2018; Bruna & Li, 2017; Monti et al., 2018). (Gilmer et al.) formulated graph neural networks in terms of message passing, and (Hamilton et al., 2017a) developed efficient mechanisms for learning on large-scale graphs. We note that most graph neural networks assume that the underlying graph is *given* and *fixed* and the graph convolution-like operations typically amount to modifying the node-wise features. Architectures like message passing neural networks (Gilmer et al.) or primal-dual convolutions (Monti et al., 2018) allow also to update the edge features, but the graph *topology* is always kept the same. This often happens to be a limiting assumption. In many problems the data can be assumed to have some underlying graph structure, however, the graph itself might not be explicitly given (Liu et al., 2012). This
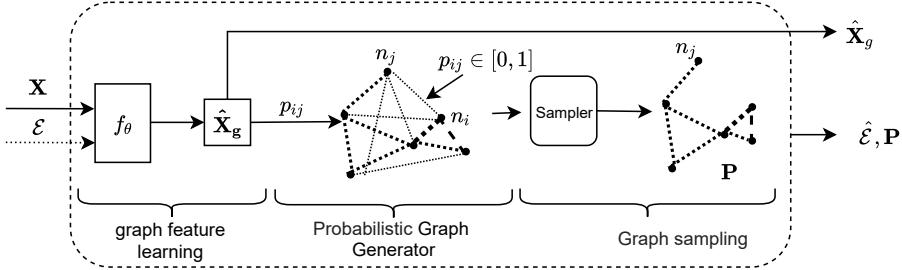
Figure 1. Details of differentiable Graph Module (DGM). It can be divided into 3 parts, a) graph feature learning f_θ, b) probabilistic graph generator detailed in eq. 1 generates a fully connected graph, c) sampler: a stochastic relaxation of k-NN rule is detailed in graph sampling.

is the case, for example, in medical and healthcare applications, where the graph may be noisy, partially- or even completely unknown, and one is thus interested in inferring it from the data. This is especially important in inductive settings where some nodes might be present in the graph at test but not training. Furthermore, sometimes the graph may be even more important than the downstream task as it conveys some interpretability of the model. Several geometric models allowing to learn the graph have recently been studied. (Kipf et al., 2018) proposed a variational autoencoder, in which the latent code represents the interaction graph underlying a physical system, and the reconstruction is based on graph neural networks. Wang et al. (2019) proposed dynamic graph CNNs for the analysis of point clouds, where a kNN graph is constructed on the fly in the feature space of the neural network. Zhan et al. (2018) proposed constructing multiple Laplacians and learn to weight them during optimization. Similarly, Li et al. (2018a) proposed a spectral graph convolutional method, in which residual Laplacian computed on the feature output from each layer and the input Laplacian is updated after each layer. Both the method learn the graph through Laplacians but still needs an initial graph. Huang et al. (2018) proposed another version of spectral filters that parametrize the Laplacian instead of the coefficient of the filter. In this paper, we propose a generalized technique for learning the graph based on the output features of each layer and optimize these graphs along with the network parameters during the training. The main obstacle for including the graph construction as a part of the deep learning pipeline that, being a discrete structure, it is non-differentiable. Inspired by (Plötz & Roth, 2018), we propose a technique that enables the backpropagation through the graph. The main idea is to use the continuous deterministic relaxation of neighborhood selection rules such as kNN, thus allowing differentiating the output w.r.t. the edges of the graph. In order to avoid the use of a pre-fixed graph, we leverage kNN graph on the input feature representation of each node, separately for each layer (Wang et al., 2019). In the subsequent sections, we describe our model

and extensively evaluate it on applications from the domains of healthcare (disease prediction), brain imaging (gender and age prediction), computer graphics (3D point cloud segmentation), and computer vision (zero-shot learning). Our model shows significant improvement over baselines both in transductive and inductive settings and achieves state-of-the-art results.

2. Method

Given a set of N input nodes and associated features $\mathbf{X} \in \mathbb{R}^{N \times d}$, our goal is to discover the underlying latent graph structure in order to enable the use of graph convolutional operators for learning classification tasks. A graph $\mathcal{G} = (\{1, \ldots, N\}, \mathcal{E})$ is by construction a discrete structure, where an edge $(i, j) \in \mathcal{E}$ linking two nodes is either present or absent. This makes $\mathcal{G}$ non-differentiable with respect to its edge set $\mathcal{E}$, therefore it cannot be directly optimized with gradient-descent based optimization techniques.

To overcome this limitation, we replace the edge set $\mathcal{E}$ with weighted adjacency $\mathbf{P}$, where $p_{ij} \in (0, 1]$ is interpreted as the probability of $(i, j) \in \mathcal{E}$. The probability p_{ij} is computed in a separate feature space $\hat{\mathbf{X}}_g = f_\Theta(\mathbf{X}_g)$ designated as the *graph representation*, where f_Θ is a learnable function. A graph constructed this way can than be sampled according to p_{ij} to be used in any graph convolutional layer for node representation learning.

In the following subsections we introduce the Differentiable Graph Module (DGM), propose a general architecture that exploits DGM for node-wise classification, and show how p_{ij} can be optimized in a task-driven fashion.

2.1. Differentiable Graph Module (DGM)

As shown in figure 1, DGM takes the node features $\mathbf{X}_g$ and the set of edges $\mathcal{E}$ (if available) as input, and outputs a new set of edges $\hat{\mathcal{E}}$. We divide the operation of DGM in three parts.
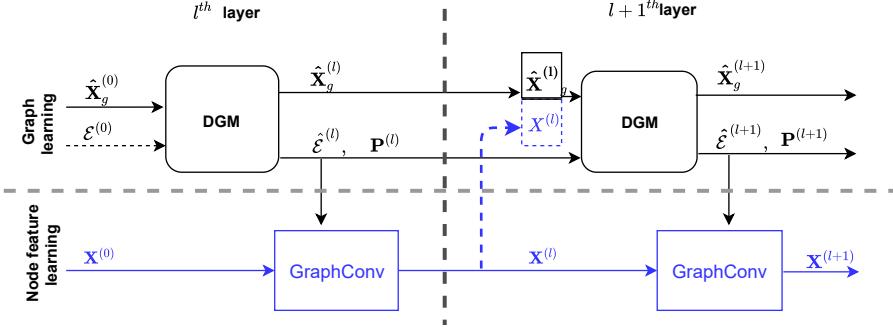
Figure 2. Forward propagation rule using the proposed DGM. We show the model architecture of two consecutive layers and the flow from input features to the predicted edges.

Graph representation feature learning. The learnable part of our DGM conists of a parametric function $\hat{\mathbf{X}}_g = f_\theta(\mathbf{X}_g)$ transforming input features $\mathbf{X}_g$ into features $\hat{\mathbf{X}}_g$ used for graph representation as explained in the next paragraph. The function f_θ could be in principle any non-linear function, such as a small neural network (MLP), or a graph convolution operator if an input graph is provided.

Probabilistic graph generator. The probabilistic graph generator part (shown in fig 1) assumes initially a fully connected graph and computes the probability

$$p_{ij} = e^{-t\|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2} \qquad (1)$$

of the edge $(i, j) \in \mathcal{E}$. Here t is a optimized temperature parameter and $\hat{\mathbf{x}}_i \in \hat{\mathbf{X}}_g$ is the output of f_θ. Such a continuous modeling of $\mathcal{E}$ allows back propagation of the gradients through the neighborhood selection.

Our choice of using a Euclidean embedding in eq. 1 for defining the edge probability reduces the complexity in comparison to an architectural choice, for instance, an MLP that takes features of two nodes as input to predict their probability (Jang et al., 2019). We note that other spaces, e.g. with hyperbolic geometry (Krioukov et al., 2010), could also be used.

Graph sampling From the estimated edge probability matrix $\mathbf{P}$, we then sample a fixed k-degree graph. We make use of the *Gumbel-Top-k trick* (Kool et al., 2019) for sampling the unnormalized probability distribution defined in equation 1, thus making the sampling a stochastic relaxation of the k-NN rule.

Let $\mathbf{p}_i = (p_{ij} : j = 1 \ldots N)$ be the unnormalized probability distribution of ingoing edges of ith node. We extract k edges according to the first k elements of argsort($\log(\mathbf{p}_i) - \log(-\log(\mathbf{q}))$) where $\mathbf{q} \in \mathbb{R}^N$ is uniform i.i.d. in the interval $[0, 1]$. (Kool et al., 2019) prove that the samples extracted this way follow the categorical distribution

$(p_{ij} / \sum_r p_{ir} : j = 1 \ldots N)$. We denote the new extracted set of edges by $\hat{\mathcal{E}}$. Finally, the DGM outputs the unweighted graph $\hat{\mathcal{G}} = (\{1 \ldots N\}, \hat{\mathcal{E}})$.

2.2. Forward propagation rule

As shown in figure 2, at each layer l, DGM is used as a block to learn the graph $\hat{\mathcal{G}}^{(l)}$, which is then passed as input to the separate 'GraphConv' operation to learn the node representations for the task at hand. We compute output features $\mathbf{X}^{(l+1)}$ at layer l as,

$$\hat{\mathbf{X}}_g^{(l+1)} = f_\theta(\mathcal{E}^{(l)}, \mathbf{X}_g^{(l)})$$
$$\mathcal{E}^{(l+1)} = \hat{\mathcal{E}}^{(l)} \sim \mathbf{P}^{(l)}(\hat{\mathbf{X}}_g^{(l+1)})$$
$$\mathbf{X}^{(l+1)} = g_\phi(\mathcal{E}^{(l+1)}, \mathbf{X}^{(l)}),$$

where ϕ and θ are learned parameters of some non-linear functions f and g. $\mathbf{X}_g^{(l)}$ are features given as input to DGM for graph representation.

In its simplest implementation $\mathbf{X}_g^{(l)} = \mathbf{X}^{(l)}$, meaning that we use same input features for both graph and node representation. We instead propose to use the concatenation of previous graph and node representation features for $l > 1$:

$$\mathbf{X}_g^{(l)} = [\hat{\mathbf{X}}_g^{(l)} | \mathbf{X}^{(l)}]. \qquad (2)$$

The final node features $\mathbf{X}^{(L)}$ of last layer L are then used to generate the predictions. Whether not specified we use 'EdgeConv' proposed by (Wang et al., 2019) for both f and g, since it is the natural choice for our fixed k-degree graph sampling:

$$\hat{\mathbf{x}}_i = \underset{j:(i,j)\in\hat{\mathcal{E}}}{\square} h_\psi(\mathbf{x}_i, \mathbf{x}_j) \qquad (3)$$

where $\square$ is the permutation-invariant aggregation operation (chosen as $\sum$ in our paper) and h_ψ is a non-linear function with a set of ψ as the learnable parameters. In the first DGM layer, where no graph is available, we just set f as the identity function, letting the network learn only the temperature parameter.

2.3. Graph optimization loss

The sampling scheme we adopt does not allow the gradient of any classification loss function involving just graph features $\mathbf{X}$ to flow through the graph prediction branch of our network. To allow its optimization we exploit tools from reinforcement learning, rewarding edges involved in a correct classification and penalizing edges that led to misclassification.

Suppose that, after a forward step, the network outputs the classification y_i for the input features $\mathbf{x_i}$ with sampled edges $\hat{\mathcal{E}}^{(\ell)}$ at layer l. We define the following graph loss function:

$$L_{graph} = \sum_i \delta(y_i, \tilde{y}_i) \prod_{l=1}^{L} \prod_{j:(i,j)\in\hat{\mathcal{E}}^{(l)}} p_{ij}^{(l)} \qquad (4)$$

, where $\delta(y_i, \tilde{y}_i)$ is a function taking value -1 if $y_i = \tilde{y}_i$ and 1 otherwise, and $\tilde{y}_i$ is the ground truth label.

The previous definition intrinsically weights unevenly positive and negative samples, especially in the early stages of the training where the classification accuracy is low. This drives the network to favor a uniform low probability estimation for all the edges. To prevent this behavior we weight positive and negative samples according to the current per-class accuracy:

$$L_{graph} = \sum_{\alpha}^{Classes} \sum_{i\in\alpha} \delta_{\alpha}(y_i, \tilde{y}_i) \prod_{l=1}^{L} \prod_{j:(i,j)\in\hat{\mathcal{E}}^{(l)}} p_{ij}^{(l),} \quad (5)$$

$$\delta_{\alpha}(y_i, \tilde{y}_i) = \begin{cases} \mathrm{acc}_{\alpha} - 1 & \text{if } y_i = \tilde{y}_i \\ \mathrm{acc}_{\alpha} & \text{otherwise} \end{cases} \qquad (6)$$

with acc_{α} being the class accuracy computed on predictions y_i. Using a per-class accuracy rather than a global accuracy helps in dealing with uneven distribution of samples among different classes in the dataset.

Graph loss L_{graph} is then optimized by summing it with the classification loss (e.g. Categorical Cross-Entropy)

2.4. Multi-modal setting

Multi-modal datasets consist of two (or more) sets of features coming from different modalities. The graph can be learned from the one of the modalities and node representation from the other modality. Towards this, we provide a variant of DGM named as 'Multi-modal GDM' (M-GDM). The only difference w.r.t. the forward propagation described above is that we train the graph learning part on separate set of features dedicated for the graph learning purpose. Thus amounts to using only $\hat{\mathbf{X}}_g^{(l)}$ in eq. 2.

2.5. Out-of-sample extension

One of the major challenges for the graph-based techniques is the out of sample extension. Spectral convolution-based methods, in particular, need a pre-defined graph. In such a setting, it is difficult to change the graph or to add the nodes or to use the filters that are learned for the input graph. In the spatial techniques, the underlying graph $\mathcal{G}$ needs to be defined beforehand. In the case of out of sample extension, the whole graph needs to be redefined incorporating the test samples.

Different methods have been proposed to solve the out of sample extension, such as (Kipf et al., 2018) and graph-SAGE (Hamilton et al., 2017a). In the graphSAGE method, an embedding function is learned based on the node features and the local neighborhood of each node. Then the unseen points are projected to the node embedding. However, the graphSage method still requires a graph. In this section, we show that our proposed method can be extended to an inductive setting

In our method, the graph is optimized with the task at hand. From equation 3 we focus on learning the function f_θ and g_ϕ, hence the learnable parameters θ and ϕ during the training. Since the graph in our case is dynamic and generated at each layer, it is easy to generate the new graph with a dynamic number of nodes as well. In the inductive setting, the parameters θ and ϕ are used to learn the representations based on the previously trained filters.

3. Experiments and Results

In this section, we show the diverse nature and superiority of our method to 4 different applications. We choose 4 datasets to cover a wide variety of possible heterogeneity in the data.

3.1. Application to disease prediction

Given multi-modal features collected in the hospital, the task is to predict the disease for each patient. Here, we target Alzheimer's disease prediction given imaging features (MRI, fMRI, PET) and non-imaging (demographics and genotypes) per patient. We pose this problem as a classification of each patient either of the 3 classes viz. Normal, Alzheimer's and Mild Cognitive Impairment (MCI).

GCNs are being leveraged to utilize such rich multi-modal data. In such a setting, the graph is constructed on the entire population where each patient is considered as a node and the connectivity between the patient is based on the similarity in their respective non-imaging features. Imaging features are assigned to each node. Finally, the features for each node are learned from this setting and used for the classification task. For this experiment we use Tadpole (Marinescu et al., 2018) dataset which is a subset of the Alzheimer's Dis-

Method	Accuracy
Linear classifier	70.22± 06.32
Multi-GCN (Kazi et al., 2019a)	76.06 ± 00.72
Spectral-GCN (Parisot et al., 2017)	81.00 ± 06.40
InceptionGCN (Kazi et al., 2019b)	84.11 ± 04.50
DGCNN	84.59 ± 04.33
M-DGM	**90.05 ± 03.70**
DGM	**91.05 ± 05.93**

Table 1. The accuracy of classification on the Tadpole dataset. We compare the proposed method with respect to the state of the art. The table proves that DGCNN is a strong baseline to compare with in the further experiments.

ease Neuroimaging Initiative (adni.loni.usc.edu), consisting of 557 patients with 354 multi-modal features per patient. Imaging features are constituted of Magnetic Resonance Imaging, Positron Emission Tomography, cognitive tests, and CSF whereas non-imaging features are constituted of demographics (age, gender), genotype and average FDG PET value.

We show three sets of experiments for this dataset. As a first experiment, in table 1 we compare the proposed method with four states of the art methods. Linear classifier represents a non-graph based method where results are obtained by ridge classifier. Multi-GCN (Kazi et al., 2019a), Spectral-GCN (Parisot et al., 2017) and InceptionGCN (Kazi et al., 2019b) are spectral approaches targeting the classification task. These three methods require a pre-defined graph obtained from non-imaging modality. We also add DGCNN as a baseline, as it dynamically builds the graph, the approach is similar to our method.

We can see from the results in a table (1) that graph-based methods perform better than the linear non-graph based method. DGCNN shows better or comparable results with respect to the spectral graph-based techniques making it a strong baseline. Both our proposed models exceed the state of the art results by 7.28% for the classification task. Further, the variance of the proposed method MM-DGM is

node-features	graph-features	DGCNN	M-DGM	DGM
M1	M1	82.98±03.35	**92.56±02.57**	89.88±03.95
M1	M2	85.65±05.90	90.05±03.70	**91.50±05.93**
M1	M1+M2	84.22±05.82	**90.96±03.59**	90.59±02.40
M1+M2	M1+M2	84.59±04.33	86.89±04.91	**90.42±03.87**
Mean		84.36±04.85	90.12±03.69	**90.60±04.04**

Table 2. The table represents the average accuracy of classification for the 10 fold cross validation in the transductive setting, for tadpole dataset. The first two columns show the feature type used for graph learning chosen between modality 1 and modality 2 corresponding to M1 and M2 respectively.

node-features	graph-features	DGCNN	M-DGM	DGM
M 1	M 1	82.99±04.91	87.94±03.02	**88.12±03.65**
M 1	M 2	81.06±04.80	87.59±03.05	**88.48±04.58**
M 1	M 1 +M 2	81.95±06.17	86.70±04.43	**89.54±05.69**
M 1 +M 2	M 1 +M 2	84.39±04.57	**88.64±03.63**	87.23±03.53
Mean		82.60±05.11	87.72±03.53	**88.34±04.36**

Table 3. The table represents the average accuracy of classification for the 10 fold cross validation in the inductive setting, for tadpole dataset. 10 % of the data is kept completely unseen.

	Gender classification		
	DGCNN	M-DGM	DGM
Transductive	87.06 ±02.89	90.00± 01.89	**90.22±02.03**
Inductive	85.31 ±06.37	88.71±03.78	**89.14±05.92**

Table 4. The table represents the accuracy of classification for gender classification task for transductive and inductive settings for the UK Biobank data

smaller than most of the state of the art methods showing the robustness of the model.

In our second experiment, shown in table 2, we vary the graph features. In this setting, we keep the node feature constant to check the sensitivity of the model towards the graph-features and compare the performance of the classification task to DGCNN. We also show the results in the inductive setting in table 3. For, this setting we keep 10% of the data completely unseen and train our model with remaining data in the regular fashion. During the inductive setting, we use the pre-trained model for the filters, while the graph in each layer is constructed over the whole population including the 10% out of sample set.

Clearly, in all the settings both our proposed models perform better than DGCNN. This means that the graph constructed by our method is better than a DGCNN (kNN selection rule). The performance of the inductive setting drops globally for all the setting on average by 2.14%. The variance of both of the proposed models is lower than DGCNN reassuring the robustness of the model.

Implementation details. As a pre-processing step, we use standard normalization for all the features and apply a dimensionality reduction technique called 'recursive feature elimination' to reduce the input feature dimension for all the experiments to 30 for all the datasets. We use $k = 5$, a learning rate of 0.01 reduced to 0.0001 at the intervals of 100 epochs in a piecewise constant fashion. We train each model for 300 epochs, optimizing the loss using Adam Optimizer. All the experiments are implemented in TensorFlow and performed using Titan Xp, 12GB GPU.

	Age prediction		
	DGCNN	M-DGM	DGM
Transductive	58.35±00.91	60.85±00.91	**61.59±01.05**
Inductive	51.84±08.16	**55.77±06.01**	53.37±07.94

Table 5. The table represents the accuracy of classification for age prediction task in transductive and inductive settings for the UK Biobank data. We divide the population into 4 groups with a bin of 10 years starting from 50 to 89 years.

3.2. Application to Gender Classification and Age Prediction Task

Similar to Tadpole dataset, we test our model for two different tasks on another dataset 'UKbiobank'. Given structural, volumetric and functional features of the brain, the first task is to predict the gender and the second to predict the age for each patient. For both tasks, we use a subsample of the UK Biobank data (Miller et al., 2016). It consists of 14,503 individuals with 440 features per individual including age and gender. The features are mainly collected from brain MRI and fMRI imaging providing the structural and functional information for each patient respectively.

Keeping the implementation details similar to Tadpole, firstly, we cast gender prediction as a binary classification task. Secondly, we devise the age prediction task as a categorical classification task. For the age prediction task, we divide the individuals into four groups based on age as group 1 (50-59 years), group 2 (60-69 years), group 3 (70-79 years) and group 4 (80-89 years) making it a four classes classification problem. We report the results for both transductive and inductive settings. For the transductive setting, we split the data into 90% training and 10% testing points whereas for the inductive setting we divide the data into 10% unseen point 80% training and 10% validation set. As can be seen from the table 4 and 5, both our models perform better than DGCNN in all the settings for both tasks.

3.3. Application to Point cloud segmentation

Point cloud part segmentation is a more challenging task from the graph optimization perspective. We are given an object represented by a set of 3D points in space with an unknown connectivity. Each object is thus a completely new set of points and there is no intersection between training and testing points.

We directly compare with (Wang et al., 2019) on the task of part segmentation of ShapeNet part dataset (Yi et al., 2016). The dataset is composed of 16881 point clouds representing 3D shapes from 16 different objects categories. Each shape's point is annotated with one of the 50 part category labels, where most of the shapes are composed by less than 6 category parts. Following the experimental setup of (Wang et al., 2019) we sample 2048 points from each training shape

	# Shapes	DGCNN	DGM
Airplane	2690	84.0	**84.1**
Bag	76	**83.4**	82.5
Cap	55	**86.7**	84.6
Car	898	77.8	**77.9**
Chair	3758	90.6	**91.3**
Earphone	69	74.7	**79.0**
Guitar	787	91.2	**92.5**
Knife	392	87.5	**87.7**
Lamp	1547	82.8	**83.7**
Laptop	451	95.7	**96.5**
Motorbike	202	66.3	**66.8**
Mug	184	94.9	**95.1**
Pistol	283	81.1	**83.1**
Rocket	66	**63.5**	62.3
Skateboard	152	74.5	**77.8**
Table	5271	**82.6**	82.2
MEAN		85.2	**85.6**

Table 6. Comparison of mIoU(%) score in ShapeNet part segmentation task.

with 3-dimensional features representing by the 3D position of the point. We follow the same train/validation/test split scheme as (Chang et al., 2015).

We mimic the same architecture used by (Wang et al., 2019) for this task, replacing their graph kNN sampling scheme by our DGM with a feature depth of 16. We keep the remaining of the network untouched, including the value of $k = 20$ and training parameters. During inference, given the stochastic nature of our graph, we repeat the classification of each point for 8 times and then choose the $argmax$ of the cumulative soft predictions.

In table 6 we report the mean Intersection-over-Union (mIoU) values calculated by averaging the IoUs of all testing shapes. Our approach allows increasing performance over the original kNN sampling scheme on almost all shape classes.

In figure 4 we show the sampling probabilities of some points (red dot) on different shapes at the last two layers of the network. We can notice that the probability of connecting two points is not related to the point feature space which is used for part classification, but it rather retains some spatial information and seems to be inspecting symmetries of the shape. Some segmentation examples are shown in figure 3.

3.4. Application to zero-shot learning task

We first define the problem of zero-shot learning and provide the details of the state-of-the-art GCN based model (Kampffmeyer et al., 2019) used for this task.
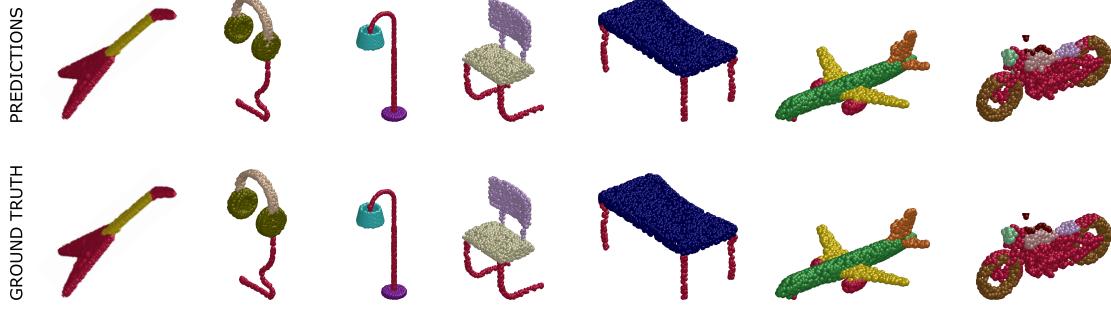
Figure 3. Segmentation examples on ShapeNet dataset. Points are colored according to their predicted (top) and ground-truth (bottom) part labels.

The problem of Zero-Shot Learning consists on the classification of samples belonging to classes that have never been seen during training phase. The most popular approach is to train a network to predict a vector representation of a category starting from some implicit knowledge, i.e. semantic embedding. The vector representation is then mapped directly to classifiers (Xian et al., 2018). Recent works showed that using also explicit relations between categories in term of knowledge graphs can help in significantly improve classification accuracy (Kampffmeyer et al., 2019; Wang et al., 2018).

Proposed Model for Zero-Shot Learning We base our model on the SGCN architecture proposed in (Kampffmeyer et al., 2019), where the input knowledge graph is replaced by our DGM module.

Let $\mathbf{X} \in \mathbb{R}^{N \times S}$ be the set of N input samples equipped with a S dimensional feature vector. In this case, each $\mathbf{X}$ is the semantic embedding (i.e. word vector) associated with
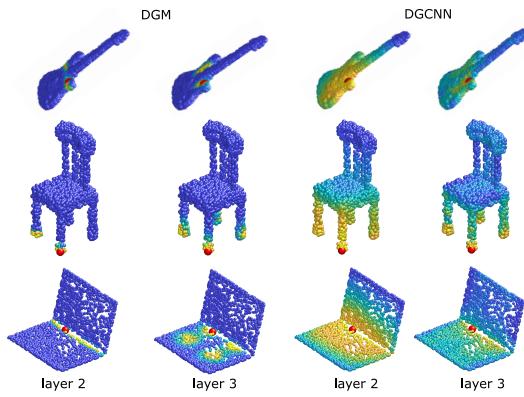

Figure 4. Comparison between our DGM (left) and original KNN sampling on the feature space (right) in the last two convolutional layers of the network. In DGM the colormap encodes the probability of each point to be connected to the red point. For the KNN sampling of DGCNN we plot the exponential of the opposite of the euclidean distance on feature space.

each category class. Each layer of the network is composed by the following convolution on graphs:

$$\mathbf{X}^{(l+1)} = \sigma \left((\mathbf{D}^{(l)})^{-1} \mathbf{A}^{(l)} \mathbf{X}^{(l)} \mathbf{\Theta}^{(l)} \right) \qquad (7)$$

where $\sigma(\cdot)$ is a *LeakyRelu* non linearity, $\mathbf{\Theta}^{(k)}$ are the learned weights and $(\mathbf{D}^{(k)})^{-1} \mathbf{A}_k$, with $\mathbf{D}_{ii}^{(k)} = \sum_j \mathbf{A}_{ij}^{(k)}$, is the non-symmetric normalization of the adjacency matrix $\mathbf{A}^{(k)}$ constructed from the graph sampled with our DGM.

The Zero-Shot task loss is thus defined as:

$$L_{task} = \sum_{i=1}^{N} \|\mathbf{w_i} - \tilde{\mathbf{w}}_\mathbf{i}\|_2^2 \qquad (8)$$

where $M < N$ is the number of training classes, $\mathbf{w_i}$ and $\tilde{\mathbf{w}}_\mathbf{i}$ are the predicted and ground-truth vector representation of the ith category.

Note that, even if in 8 we deal with a regression problem, it is straightforward to adapt it to deal with our graph loss defined in equation 6, considering $argmin_j \|\mathbf{w}_i - \mathbf{w}_j\|_2$ as the predicted category for sample $\mathbf{X}_i$.

Dataset and training details As in (Kampffmeyer et al., 2019), we use weights of the last fully connected layer of a ResNet-50 (He et al., 2016) pre-trained on ImageNet 2012 dataset (Deng et al., 2009) as our target vector representation $\tilde{\mathbf{y}}_i \in \mathbb{R}^{2049}$. Input semantic features $\mathbf{x_i} \in \mathbb{R}^{300}$ are extracted with GloVe text model (Pennington et al., 2014) trained on Wikipedia dataset. Our model consists of two graph convolution layers with hidden and output layer of

Model	ACC (%)
GCNZ	70.5
DGP	77.3
DGM (ours)	73.0

Table 7. Classification accuracy for unseen classes on AWA2 dataset. GCNZ and DGP results are reported from (Kampffmeyer et al., 2019).
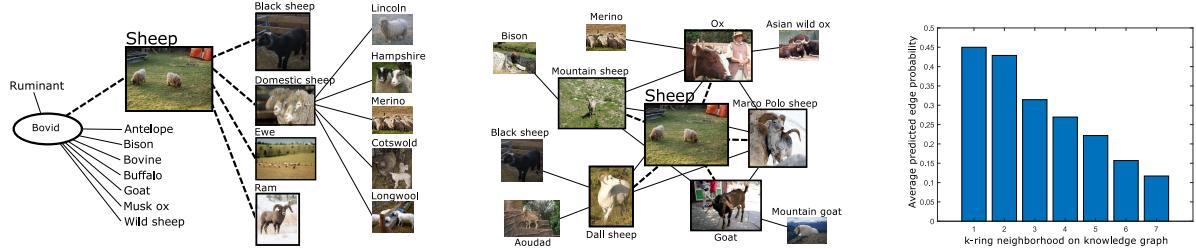
Figure 5. First two figures show an example of the 2-ring neighborhood of the "sheep" category. Left figure corresponds to the knowledge graph while central figure is our graph, sampled considering the 5 most probable edges. On the right a plot showing the average predicted probability of edges belonging to the k-ring neighborhood (AwA2 test categories). Higher probabilities corresponding to nearest neighbors suggest that the predicted graph structure is loosely related to the knowledge graph.

dimension 2048 and 2049, paired with two DGM layers of dimension 16 for graph representation and $k = 3$.

We train our model on the 21K ImageNet dataset categories, where we have as input the semantic embedding for all categories, but only the first 1K have a corresponding ground-truth vector representation. The model is trained for 5000 iterations on a randomly subsampled set of 7K categories containing all the 1K of training.

Testing is performed on AWA2 dataset, composed by 37,322 images belonging to 50 different animal categories. In table 7 we report top-1 accuracy results for the test split proposed in (Wang et al., 2018) composed by images from 10 classes not present in the first 1K of ImageNet used for training. Note that, as opposed to both GCNZ (Wang et al., 2018) and DGP (Kampffmeyer et al., 2019), we do not make use of the knowledge graph. As shown in figure 5, the knowledge graph seems indeed a good graph representation for zero-shot task but even if our predicted graph shows some similarity to it, our sampling scheme fails in capturing its hierarchical structure.

4. Discussion and conclusion

4.1. Conclusion

In this paper, we tackled the challenge of graph learning in convolutional graph neural networks. We have proposed a novel Differentiable Graph Module (DGM) that predicts a probabilistic graph, allowing a discrete graph to be sampled accordingly in order to be used in any graph convolutional

operator. Further, we devised a weighted loss inspired by reinforcement learning which allows the optimization over edge probabilities.

Our DGM is generic and adaptable to any graph convolution based method. We prove this by using our method to solve a wide variety of tasks starting from application in healthcare (disease prediction), brain imaging (age and gender prediction), computer graphics (3D point cloud segmentation) and computer vision (zero-shot learning), dealing with multi-modal datasets and inductive settings. Table 8 shows the wide heterogeneity captured by the choice of our datasets and tasks.

4.2. Discussion

There are some open problems with the proposed method. Computation-wise our method, even being more lightweight of a full pairwise MLP approach (Jang et al., 2019), still needs the computation of all pairwise distances, making it quadratic with respect to input nodes. Restricting the computation of probabilities in a neighborhood of the node and using tree-based algorithm could help in reducing the complexity to $\mathcal{O}(n \log n)$. Further, our choice of sampling k neighbors does not consider the heterogeneity of the graph in terms of the degree distribution of nodes. Other sampling schemes (e.g. threshold-based sampling (Jang et al., 2019)) could be investigated. It would be also interesting to take into consideration previous knowledge about the graph, as for instance impose a node degree distribution or even deal with an initial input graph to be optimized for a specific task.

Dataset	Tadpole	UKbiobank	ShapeNet	Awa
Multi-modal	yes	no	no	no
Sample size	557	14.5k	2048	21K
feature size	354	440	3	300
number of graphs	1	1	16881	1

Table 8. The chosen datasets show wide variety of challanges in within them

References

Battaglia, P. W. et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

Bianchi, F. M., Grattarola, D., Alippi, C., and Livi, L. Graph neural networks with convolutional arma filters. *arXiv:1901.01343*, 2019.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017.

Bruna, J. and Li, X. Community detection with graph neural networks. *Stat*, 1050:27, 2017.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Choma, N. et al. Graph neural networks for icecube signal classification. In *Proc. ICMLA*, 2018.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, pp. 3844–3852, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*. Ieee, 2009.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, pp. 2224–2232, 2015.

Gainza, P., Sverrisson, F., Monti, F., Rodola, E., Bronstein, M. M., and Correia, B. E. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17:184–192, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proc. ICML*.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Proc. NIPS*, 2017a.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv:1709.05584*, 2017b.

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

Huang, W., Zhang, T., Rong, Y., and Huang, J. Adaptive sampling towards fast graph representation learning. In *NeurIPS*, pp. 4558–4567, 2018.

Jang, S., Moon, S., and Lee, J. Brain signal classification via learning connectivity structure. *CoRR*, abs/1905.11678, 2019. URL http://arxiv.org/abs/1905.11678.

Kampffmeyer, M., Chen, Y., Liang, X., Wang, H., Zhang, Y., and Xing, E. P. Rethinking knowledge graph propagation for zero-shot learning. In *CVPR*, pp. 11487–11496, 2019.

Kazi, A., Shekarforoush, S., Kortuem, K., Albarqouni, S., Navab, N., et al. Self-attention equipped graph convolutions for disease prediction. In *ISBI*, pp. 1896–1899. IEEE, 2019a.

Kazi, A., Shekarforoush, S., Krishna, S. A., Burwinkel, H., Vivar, G., Kortüm, K., Ahmadi, S.-A., Albarqouni, S., and Navab, N. Inceptiongcn: Receptive field aware graph convolutional network for disease prediction. In *IPMI*. Springer, 2019b.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. *arXiv:1802.04687*, 2018.

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

Kondor, R. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv:1803.01588*, 2018.

Kool, W., van Hoof, H., and Welling, M. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. *CoRR*, abs/1903.06059, 2019. URL http://arxiv.org/abs/1903.06059.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguná, M. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Levie, R., Monti, F., Bresson, X., and Bronstein, M. M. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Trans. Signal Process.*, 67(1):97–109, 2018.

Li, R., Wang, S., Zhu, F., and Huang, J. Adaptive graph convolutional neural networks. In *AAAI*, 2018a.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018b.

Liu, W., Wang, J., and Chang, S.-F. Robust and scalable graph-based semisupervised learning. *Proc. IEEE*, 100 (9):2624–2638, 2012.

Marinescu, R. V., Oxtoby, N. P., Young, A. L., Bron, E. E., Toga, A. W., Weiner, M. W., Barkhof, F., Fox, N. C., Klein, S., Alexander, D. C., et al. Tadpole challenge: Prediction of longitudinal evolution in alzheimer's disease. *arXiv preprint arXiv:1805.03909*, 2018.

Mellema, C., Treacher, A., Nguyen, K., and Montillo, A. Multiple deep learning architectures achieve superior performance diagnosing autism spectrum disorder using features previously extracted from structural and functional mri. In *2019 IEEE ISBI)*, pp. 1891–1895. IEEE, 2019.

Miller, K. L., Alfaro-Almagro, F., Bangerter, N. K., Thomas, D. L., Yacoub, E., Xu, J., Bartsch, A. J., Jbabdi, S., Sotiropoulos, S. N., Andersson, J. L., et al. Multimodal population brain imaging in the uk biobank prospective epidemiological study. *Nature neuroscience*, 19(11):1523, 2016.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, 2017.

Monti, F. et al. Dual-primal graph convolutional networks. *arXiv:1806.00770*, 2018.

Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Moreno, R. G., Glocker, B., and Rueckert, D. Spectral graph convolutions for population-based disease prediction. In *MICCAI*, pp. 177–185. Springer, 2017.

Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., and Rueckert, D. Disease prediction using graph convolutional networks: Application to autism spectrum disorder and alzheimer's disease. *Med Image Anal*, 48:117–130, 2018.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP*, pp. 1532–1543, 2014.

Plötz, T. and Roth, S. Neural nearest neighbors networks. In *NeurIPS*, pp. 1087–1098, 2018.

Qi, S., Wang, W., Jia, B., Shen, J., and Zhu, S.-C. Learning human-object interactions by graph parsing neural networks. In *ECCV*, pp. 401–417, 2018.

Qi, X., Liao, R., Jia, J., Fidler, S., and Urtasun, R. 3d graph neural networks for rgbd semantic segmentation. In *ICCV*, pp. 5199–5208, 2017.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw*, 20(1):61–80, 2008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv:1710.10903*, 2017.

Wang, X., Ye, Y., and Gupta, A. Zero-shot recognition via semantic embeddings and knowledge graphs. In *CVPR*, pp. 6857–6866, 2018.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5):146, 2019.

Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE PAMI*, 2018.

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 35(6):1–12, 2016.

Zhan, K., Chang, X., Guan, J., Chen, L., Ma, Z., and Yang, Y. Adaptive structure discovery for multimedia analysis using multiple features. *IEEE transactions on cybernetics*, 49(5):1826–1834, 2018.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Proc. NeurIPS*, 2018.

Zitnik, M., Agrawal, M., and Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.