Graph Learning Networks

Vedran Hadziosmanovic [1] **Yongbin Li** [2] **Xiao Liu** [2] **Stuart Kim** [3] **David Dynerman** [1] **Loic Royer** [1]

Abstract

Graph Neural Networks (GNNs) are an architecture naturally suited to graph structured data. In this work, we consider datasets without an explicitly defined graph, but which contain some implicit relational structure. To apply GNNs to such data one must first construct a graph. We propose *Graph Learning Networks* (GLNs), which extend the GNN architecture by learning task-specific graphs, and consider the generic problem of assigning unique labels to each point of a point-set. We apply GLNs to labelling the cell nuclei of *C. elegans* nematodes from their 3D positions, and show how this outperforms previous approaches. Furthermore, we conduct experiments that show that graph learning in GLN is key to solving hard synthetic labelling problems with an implicit and non-trivial relational structure.

1. Introduction

Convolutional neural networks (CNNs) have had great successes in computer vision. How to extend these results beyond image data remains an open question. One approach is to create new deep neural networks for data structures of interest such as graphs, sets, and point clouds (Battaglia et al., 2018; Zaheer et al., 2017; Qi et al., 2017b).

Graph and set learning problems have primarily been considered separately from each other, however many of the issues with developing deep learning models for graphs are also present for sets. For example, neural networks operating on either must be invariant to the order with which set elements are provided to the network. Moreover, sets will sometimes have a useful underlying graph structure, suggesting that a connection between graph and set learning may be helpful.

Graph Neural Networks (GNNs) (Battaglia et al., 2018; Hamilton et al., 2017; Kipf & Welling, 2016; Xu et al., 2018) are a powerful model capable of exploiting known graph structures to learn useful vector representations of nodes. Most GNN approaches make use of a *message passing* procedure (Gilmer et al., 2017), which involves progressively propagating and aggregating information along the graph. These graph structures however must be explicitly given as input to GNNs. Such a graph structure is not a priori available for common data types, including 3D point-sets in which an underlying relational structure (e.g., distances between points) is present, but is not explicitly given.

In order to apply GNNs to problems without pre-specified graphs, we propose *Graph Learning Networks* (GLNs). GLNs extend GNNs by learning an ensemble of graphs describing implicit relationships. This graph generating process is trained end-to-end together with a standard GNN. Our design is motivated by a problem in biological microscopy where every point of 3D point cloud representing cells needs to labelled with its unique identity. Existing automatic approaches are limited to a restricted subset of cells (Long et al., 2009; Kainmueller et al., 2014). Applying deep learning to this problem represents a major leap forward.

This data-source is challenging because it is not possible to identify nuclei based solely on their individual features. Rather, trained biologists recognize known *constellations* of cells. Our goal is to emulate this capability and learn to generate the needed constellations (represented as graphs).

In summary, the key contributions of this paper are:

1. we introduce the problem of applying GNNs to set data and propose a generic method for learning to generate graphs jointly with the GNN operating on them and,

2. we demonstrate this approach on a real biological dataset of scientific interest where there is no a priori graph structure. We find that our method significantly outperforms the current state of the art.

2. Related Works

2.1. Set Learning

Previous work has been done on learning with sets without the explicit relational model of GLNs. Instead, these

[1]Chan Zuckerberg Biohub, San Francisco, California, USA [2]Tsinghua University, Beijing, China [3]Stanford University, Stanford, California, USA. Correspondence to: David Dynerman <david.dynerman@czbiohub.org>, Loic Royer <loic.royer@czbiohub.org>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data

methods focus on building universal set function approximators (Zaheer et al., 2017) or adapting existing deep learning methods, such as RNNs to parse sets (Vinyals et al., 2015). PointNet and its successor, PointNet++, exploit some of the specific structures commonly found in very dense point-sets (Qi et al., 2017a;b). These approaches do not explicitly model relational structure, making them very general, but also not well suited to problems based on local relations as opposed to global properties.

2.2. Graph Neural Networks

Currently, Graph Neural Networks are the state of the art for deep learning on graphs (Battaglia et al., 2018). We build GLNs off the *message passing* approaches of (Hamilton et al., 2017; Xu et al., 2018) that produce node embeddings over T rounds. During every round, we aggregate information from the neighborhood of each node using a set function (e.g. mean, sum). This produces a vector called the *message*. Each node's vector is then modified by an *update* function (e.g. concatenation of message and previous embedding). After the T rounds, a task-specific *readout* function predicts a node attribute based on the vectors in the final round. Many variants exist and are explored in (Gilmer et al., 2017; Battaglia et al., 2018).

2.3. Learning a Graph

An important extension to the above framework is that of *Graph Attention Networks (GATs)* which bring attention mechanisms to Graph Neural Networks (Veličković et al., 2018). Our work contributes to this discussion by proposing a general view that uses attention mechanism-like procedures to build a graph without using specific hand derived features or fixed kernels (Zhang & Rabbat, 2018).

Neural Relational Inference (Kipf et al., 2018) learns a latent graph structure via use of a Variational Autoencoder (VAE) (Kingma & Welling, 2013) where the latent variables identify whether an edge connects a pair of nodes. This latent structure is then decoded by a Graph Neural Network. This work is a major step toward learning a task specific graph. However, the use of a VAE results in training difficulties which our approach avoids by using a relatively simple graph generator. Other generative models over graphs have been studied as well (You et al., 2018; Simonovsky & Komodakis, 2018; Johnson, 2017; Li et al., 2018). Generative methods offer an alternative to attention mechanisms as a method of producing graphs, but currently these methods suffer from scaling issues, strong assumptions on the training data available, or have complex training procedures. Nonetheless, they represent an important direction for future work. Another interesting direction of research is the application of transfer learning to producing a graph, which in then used in multiple tasks (Yang et al., 2018).

3. Our Approach

We consider the problem of labelling or classifying each point of a point-set. Importantly, we consider a class of hard problems for which these feature vectors considered one-by-one are not sufficient for labeling. Instead, labelling of a node requires considering other nodes, their feature vectors, and their relationships.

Our model consists of two phases, see Fig. 1. First, a graph generator takes the point-set and generates a graph. Second, this graph is given as input to a round of message passing (Gilmer et al., 2017). This process is then repeated for T rounds, with each round relying on a distinct graph generator.

3.1. Learning to generate a weighted graph

Suppose that for a given message passing round $t \in [0, T-1]$ we are given a vector embedding $\mathbf{h}_i^t \in \mathbb{R}^D$ for each node i, where D is a hyperparameter.

We generate a graph for our input point set by learning edge weights α_{ij}^t from the node features $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$ via an adjacency matrix generator ϕ:

$$\alpha_{ij}^t = \phi(\mathbf{h}_i^t, \mathbf{h}_j^t) \tag{1}$$

We propose using the inner product between the feature vectors embedded in a higher dimensional space as ϕ, although we test other variants. Formally, we have:

$$\phi(\mathbf{h}_i^t, \mathbf{h}_j^t) = \langle f^t(\mathbf{h}_i^t), f^t(\mathbf{h}_j^t) \rangle \tag{2}$$

where $f^t : \mathbb{R}^D \to \mathbb{R}^E$ is a multi-layer perceptron (MLPs) with leaky RELU activation, and $\langle \, , \, \rangle$ is the Euclidean inner product on $\mathbb{R}^E$, with E a hyperparmeter. We normalize these α_{ij}^t by applying a softmax over all j. In reference to this form being known as Inner Product Similarity (Okuno et al., 2018), we call our architecture **GLN-IPS-SG**.

3.2. Message Passing with a Learned Graph

Once we have computed an adjacency matrix $[\alpha_{ij}^t]$ for a given round t, we are able to apply any one of several standard Graph Neural Network architectures that require a graph as input. We find that a GNN built by combining ideas from (Hamilton et al., 2017; Xu et al., 2018; Kipf et al., 2018), as we describe next, performs best for our task.

Given a vector $\mathbf{h}_i^t$ for node i at round t, message passing across the graph is performed by computing $\mathbf{h}_i^{t+1}$ as:

$$\tilde{\mathbf{h}}_i^t = r^t(\mathbf{h}_i^t)$$

$$\boldsymbol{\eta}_i^t = \sum_{j=1}^{N} \alpha_{ij}^t \tilde{\mathbf{h}}_j^t \tag{3}$$

$$\mathbf{h}_i^{t+1} = \mathbf{h}_i^t + s^t(\mathbf{h}_i^t + \boldsymbol{\eta}_i^t), \tag{4}$$

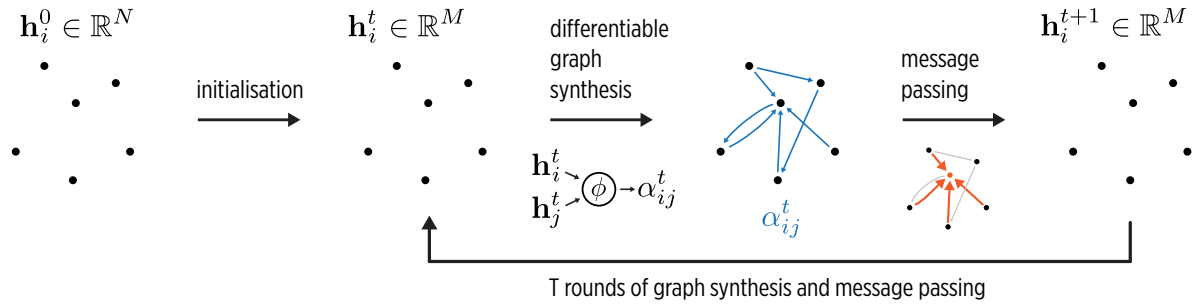

Figure 1. Graph Learning Networks (GLN). We perform T rounds of graph synthesis and message passing on a set of vectors $\mathbf{h}_i^0$ of dimension N. Graph synthesis is done by applying *adjacency matrix generator* ϕ for every pair of feature vectors: $\phi(\mathbf{h}_i^t, \mathbf{h}_j^t)$.

where $r^t, s^t : \mathbb{R}^D \rightarrow \mathbb{R}^D$ are MLPs with ELU activation (Clevert et al., 2015). We use a residual scheme (equation 4) instead of a direct assignment to update our embeddings (Kipf et al., 2018; Xu et al., 2018).

Note that when we compute the updated embedding $\mathbf{h}_i^{t+1}$ for node i, vectors for other nodes j only appear in the summation in equation 3. Since this summation is invariant under permutation of indices, the produced vectors are equivariant under reordering of the input set. We also consider some alternative methods, mainly for ablation.

3.2.1. Adjacency matrix generators variants

We consider modifying equation 2 to:

$$\phi(\mathbf{h}_i^t, \mathbf{h}_j^t) = \text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}^t\mathbf{h}_i^t, \mathbf{W}^t\mathbf{h}_j^t]) \quad (5)$$

With $\mathbf{a}, \mathbf{W}$ learned. We term this variant **GLN-GAT**, reflecting the fact that this was the attention mechanism used in Graph Attention Networks (Veličković et al., 2018). We also consider 3 variants with no learned parameters to compare against a naive, direct applications of GNNs to sets:

$$\phi(\mathbf{h}_i^t, \mathbf{h}_j^t) = \langle \mathbf{h}_i^t, \mathbf{h}_j^t \rangle \quad (6)$$

Where $\langle\ ,\ \rangle$ is the Euclidean inner product on $\mathbb{R}^D$. We term this variant **GNN-DOT** because we are passing an adjacency matrix of dot products to a standard GNN architecture. In a similar vein, we consider **GNN-L2**, where a GNN operates on an adjacency matrix of the L2 distances:

$$\phi(\mathbf{h}_i^t, \mathbf{h}_j^t) = ||\mathbf{h}_i^t - \mathbf{h}_j^t||_2 \quad (7)$$

Lastly, we consider a **GNN-UNI** variant which just assigns every node the same *uniform* weight, that is:

$$\phi(\mathbf{h}_i^t, \mathbf{h}_j^t) = 1 \quad (8)$$

GNN-UNI is actually a direct application of GNNs to sets that encodes a belief that all nodes are equal and pass messages over a fully-connected graph. It corresponds to learning a representation of the set as a whole without acknowledging substructure and thus falls into the category of set learning methods discussed in section 2.

3.3. Readout layer

Finally, after T rounds of both graph generation and message passing, we take the vector $\mathbf{h}_i^T$ from the final round of message passing apply a fully connected softmax-layer that classifies each node. We train our network using the Adam optimizer (Kingma & Ba, 2014) on cross entropy as our loss function, calculated over every point in every set.

3.4. Multi-graph learning

Similar to (Veličković et al., 2018; Kipf et al., 2018) we consider having K parallel weighted graphs instead of just one per round. Instead of computing a single α_{ij}^t for each pair of nodes i, j, we compute K weights $\alpha_{ij,k}^t$, average the messages over them, and update the embeddings as before using equation 4. We denote the multi-graph version of our method as **GLN-IPS-MG**. Formally:

$$\alpha_{ij,k}^t = \langle f_k^t(\mathbf{h}_i^t), f_k^t(\mathbf{h}_j^t) \rangle$$
$$\boldsymbol{\eta}_i^t = \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{N} \alpha_{ij,k}^t r_k^t(\mathbf{h}_i^t)$$

4. Main Results

4.1. Problem statement

We apply GLNs to a point-set labelling problem arising in *C. elegans* biology. We train each model on 180 worm point clouds 5 times, use a validation set of 20 images to select hyper-parameters, including early-stopping time, and then test the best run on 200 previously unseen point clouds divided into 10 sets of 20 and report the mean accuracy over

Table 1. Mean performance summary of different approaches on the *C. elegans* dataset. Standard deviation in parentheses. Each model was trained 5 times and the best results were reported.

METHOD	TEST ACCURACY
DEEPSET	47.4%(2.8)
POINTNET	77.2%(0.2)
GNN-L2	69.6%(3.0)
GNN-DOT	78.0%(1.9)
GNN-UNI	79.3%(1.7)
GLN-GAT	78.7%(1.6)
GLN-IPS-SG	83.7%(1.7)
GLN-IPS-MG ($K = 2$)	**84.8%(1.4)**
GLN-IPS-MG ($K = 8$)	82.1%(1.3)

these sets as well as standard deviation in table 1. This testing procedure is motivated by the size of typical *C. elegans* experiments and is meant to provide an estimate for what level of performance a *C. elegans* researcher could expect from applying our trained network to their problem. Our results are summarized in table 1.

4.2. Contextualizing the Results

Table 1 shows several variants of **GLN-IPS** outperforming state-of-the-art methods PointNet[1] (Qi et al., 2017a) and DeepSet (Zaheer et al., 2017). We believe this is due to the explicit relational modeling our method has, a bias DeepSet and PointNet as generic universal set function approximators do not have. This claim is substantiated by the fact that our baseline methods which represent naive applications of GNNs to sets without a graph learning stage get similar performance to DeepSet and PointNet. We further explore this claim with a synthetic example in section 5.

A direct comparison to leading approaches is difficult due to these methods considering only a sub-problem defined on 357 out of the 558 nuclei present. Furthermore, since they match to a manually constructed template, the number of nuclei per point-set must be exactly the number of nuclei in the template (Long et al., 2009; Kainmueller et al., 2014). Despite tackling a larger problem, our approach achieves accuracy levels on the whole problem only slightly lower than the accuracies these template methods report on the simpler sub-problem. Furthermore, these methods used features specific to *C. elegans* whereas our deep learning approach is application agnostic.

4.3. Interpreting the learned graphs

To better understand the learned graphs, we examine the top 8 strongest edges between a given cell and all other cells in the adjacency matrix $[\alpha_{ij}^t]$ after the first round of graph generation. Interestingly, we find connections to known biology. For example, a similar set of 8 nuclei are the strongest partners for nearly all other nuclei. Moreover, these groups seem to cluster around the dense anterior part of the worm, but also at the very tip of the worm's tail. This suggests that in this first round of graph generation, the GLN uses different *anchors* to produce the first high-dimensional embeddings. We also examine the distribution of errors throughout the worm. Among the most misidentified nuclei we find a series of nuclei from the hypodermal syncytium and SABD – both regions are known to be highly complex because of lineage variations and high cell density.

[1]PointNet's successor, PointNet++ (Qi et al., 2017b) is not benchmarked against because its addition of sampling methods is not applicable in our case.

Table 2. Comparison of a relational approach to a set-based approach as the number of data points increases. Each model was trained 5 times and the best results were reported.

METHOD	N = 16	N = 32	N = 48
DEEPSET	100.0%	60.0%	41.7%
GNN-UNI	86.3%	44.8%	30.2%
GLN-IPS	**100.0%**	**99.4%**	**81.7%**

5. Additional Experiment

As DeepSet is a universal set function approximator, in principle, GLN-IPS should offer no benefit. An important question is whether GLN-IPS has a bias that allows it to learn representations that generalize better in practice. We design a node identification task (named *Flip*) to answer this question affirmatively.

We consider N points $\{P_i\}$ on a line with midpoint M. Each node is given a unique label c that corresponds to a unique distance to M. However, it is randomly flipped about M in each example, meaning each node can be in two different positions along the line. This gives a total of 2^N possible input sets. This combinatorial explosion hides the simple fact that the relationship (distance) between P_i and M is all that is needed for unique identification.

As we vary N from 16 to 50 while holding the training set size constant at 36, we see that the performance of non-relational methods (DeepSet, GNN-UNI) drops dramatically while the relational bias of GLN-IPS allows its performance to remain high (table 2).

6. Conclusion

In this work, we extend Graph Neural Networks (GNNs) to operate on data without an explicitly defined graph structure. Our work generates graphs in a task specific way. This method outperforms set based methods that look at global properties at a real cellular identification task.

7. Acknowledgements

The authors would like to thank Josh Batson for his invaluable conversations, Anitha Krishnan for her help in the earliest days of this project, and Jim Karkanias for his mentoring and support.

References

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. Relational inductive biases, deep learning, and graph networks, 2018.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry, 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs, 2017.

Johnson, D. D. Learning graphical state transitions. 2017.

Kainmueller, D., Jug, F., Rother, C., and Myers, G. Active graph matching for automatic joint segmentation and annotation of c. elegans. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 81–88. Springer, 2014.

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural Relational Inference for Interacting Systems. *ArXiv e-prints*, February 2018.

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

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs, 2018.

Long, F., Peng, H., Liu, X., Kim, S. K., and Myers, E. A 3d digital atlas of c. elegans and its application to single-cell analyses. *Nature methods*, 6(9):667, 2009.

Okuno, A., Kim, G., and Shimodaira, H. Graph embedding with shifted inner product similarity and its improved approximation capability. *CoRR*, abs/1810.03463, 2018.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017a.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pp. 5099–5108, 2017b.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

Vinyals, O., Bengio, S., and Kudlur, M. Order Matters: Sequence to sequence for sets. *ArXiv e-prints*, November 2015.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yang, Z., Zhao, J., Dhingra, B., He, K., Cohen, W. W., Salakhutdinov, R., and LeCun, Y. Glomo: Unsupervisedly learned relational graphs as transferable representations, 2018.

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. Graphrnn: Generating realistic graphs with deep auto-regressive models, 2018.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3391–3401. Curran Associates, Inc., 2017.

Zhang, Y. and Rabbat, M. A graph-cnn for 3d point cloud classification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6279–6283. IEEE, 2018.