Multi-Task Learning on Graphs with Node and Graph Level Labels

Chester Holtz*
University of California, San Diego
La Jolla, CA 92093, USA
chholtz@eng.ucsd.edu

Onur Atan **Ryan Carey** **Tushit Jain**
Qualcomm Technologies, Inc.
San Diego, CA 92121 USA

Abstract

While numerous graph machine learning architectures are available for supervised learning using either node labels or graph labels, one unexplored area is learning on graphs from hierarchical labels. For example, to encode traffic patterns in a road network, one may want to take into account both per-road (node) as well as per-town (subgraph) usage metrics. When encoding the graph structure of a molecule, it may be useful to include both local measurements (e.g., bond lengths) as well as per-molecule (e.g., solubility). We have developed a technique for supervised learning using both levels of labels that uses a multi-task architecture. We validate the performance of our framework on the Proteins and Enzymes datasets and demonstrate strong performance of our algorithm compared to several benchmark methods for both graph-level and node-level prediction tasks.

1 Introduction

Learning on graph-structured data is a well-studied problem, and recent advances in graph neural networks have produced promising results. A wide variety of techniques for learning representations of graphs have been applied to a diverse set of problems including social network analysis, drug discovery, and molecular structure analysis [1, 2, 3] in both fully supervised and semi-supervised settings [4].

A typical prerequisite for hierarchical or multi-scale-based methods is a scale-hierarchy which is either pre-defined or learned from the data. In this paper, we propose to induce multi-scale representation learning on graphs with neural networks through a multi-task learning framework which simultaneously optimizes local and global losses and backpropogates error signals from multiple levels of the scale-hierarchy. The levels of the scale-hierarchy are learned through graph pooling layers which progressively coarsen, or cluster, nodes of the input graph while classification layers can either be learned for each level of the hierarchy or coupled over all layers. By propagating local and global losses, we hypothesize two advantages: (i) by increasing the number and diversity of error signals observed by the neural network, we facilitate convergence to better solutions (ii) by encouraging the network to learn both global and local properties of the data, we facilitate representations that generalize better to unseen data.

We validate our proposed method on several public datasets which are labeled at the node and graph level. Resources which provide comprehensive sets of links for graph datasets - including datasets that contain labels at different scaless include [5, 6].

*corresponding author

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

1.1 Related Work

Multi-scale and Hierarchical Learning Hierarchical, or multi-scaled, analysis of structured data has been successfully applied in many application domains. Most existing multi-scale neural network-based methods derive multi-scale representations by considering the feature maps associated with different levels of the network directly [7, 8].

Similar to our work, [9] proposed the Hierarchical Multi-label Classification Network which learns representations by jointly optimizing global and local loss functions.

Multi-task Learning Machine learning models are typically trained to optimize a particular metric which corresponds to predictive performance on a single task. However, by learning to do well on only a single task, the trained model may ignore relevant information that might assist in generalizing predictions. Specifically, we can learn on training signals for multiple related tasks.

More technically, multi-task learning can help improve the generalization of models by introducing an inductive bias. For example, a classical form of inductive bias is ℓ-p norm regularization [10] which leads to a preference for solutions with certain structures (e.g. ℓ-1 regularization results in sparse solutions). In the case of multi-task learning, the inductive bias is induced through the joint training of auxiliary tasks - in our cases tasks corresponding to prediction at different levels of label hierarchy - which results in the model preferring representations that effectively capture both local and global information. The efficacy of multitask learning has been demonstrated on a variety of tasks in NLP and computer vision [11, 12].

Deep Learning on Graphs Recent work on graph-learning has focused on techniques for learning effective node-level representations. Methods to learn effective representations include Graph Convolutional Networks [4] and their extensions including Graph Attention Networks [13] and GraphSAGE [13]. A significant drawback of these methods are their inability to propagate hierarchical information to facilitate learning subgraph or graph representations - e.g. in support of graph or subgraph-level classification. To remedy this, [1] proposed to compose node-representations into subgraph representations via summation, [14] proposed to introduce "virtual" nodes that are connected to sets of nodes, and [15] proposed a deep sets-based approach. Learning deterministic hierarchies prior to classification based on the network structure has also been explored [16, 17]. Recently, more sophisticated graph pooling operators have been proposed including DiffPool [18], SAGPool [19], and Edge Contraction Pooling [20]. These methods seek to pool node-level representations into coarser subgraph-representations through weighted aggregations, where the weights are typically learned from the data and network structure. An empirical review and comparison of different graph convolution & pooling operators and the sensitivity of graph neural networks to weight initialization is given in [21].

Our framework is comprised of graph convolutional layers and hierarchical pooling layers, and we show that formulating multi-scale graph representation learning as a multitask learning problem can be an effective method for improving performance on whole-graph, or subgraph-level classification - particularly in the case where the number graph-level instance labels is significantly fewer than the number of node-level labels.

1.2 Contribution

In summary, our contribution includes the following: (i.) We propose a general multi-scale graph embedding framework, which, to the best of our knowledge, is the first one that considers node-level and subgraph-level labels. (ii.) We demonstrate that our network outperforms strong graph convolution-based baselines on graph and node classification benchmarks.

2 Proposed Method

2.1 Preliminaries

We define a graph $G = (V, E)$ where V is the set of n nodes and E is the set of edges. Denote $X \in \mathbb{R}^{n \times d}$ to be the node-level feature matrix and $A \in \{0, 1\}^{n \times n}$ is the adjacency matrix. For a given graph $G = (V, E)$, we are given labeled nodes (local measurements) $y = [y_v]_{v \in V}$ and labeled graphs (macro measurements) z. Denote the data set $\mathcal{D} = ((G_1, y_1, z_1), \ldots, (G_m, y_m, z_m))$.

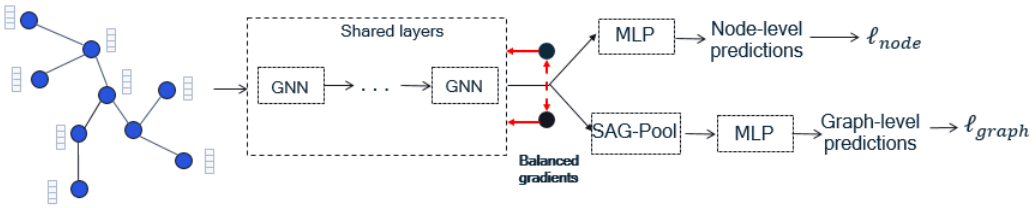

Figure 1: Neural Network Structure

We'd like to note that node-level labels have some relationship with the graph-measurements but the aggregation function from the node-level labels to graph-level labels are not known a priori in the most of the applications. Our goal is to learn a mapping from graphs to the set of the labels (both node-level and graph-level), that is $f : \mathcal{G} \rightarrow \mathcal{Y} \times \mathcal{Z}$.

In this work, we build upon GNNs and SAGPool to learn the mapping function.

Graph Neural Networks are used to learn node embedding based on the node-features and connectivity structure. We adopt the convolution formulation of [4]. The iterative message passing scheme is given by

$$H^k = M(A^{k-1}, H^{k-1}, \Theta^{k-1}) \tag{1}$$

where M is the message-passing function, Ak is a matrix that characterizes the graph structure at the kth iteration, H^k are the node embeddings, and Θ^k are the parameters of the neural network.

SAGPool is the pooling operation on the graphs. The idea is to select a reduced number of the nodes based on output of the GNN layer with self-attention weights. Let k is the pre-defined ratio for the node selection. Then, the nodes ids of those that are selected are given by

$$\text{idx} = \text{top-rank}(Z, \lceil kN \rceil) \tag{2}$$

where Z is an output of a GNN layer with self-attention weights, that is $Z = \sigma(GNN(A, X))$.

Grad-norm [22] tunes the weights of the graph-level prediction loss and node-level prediction loss to makes imbalanced gradient norms similar.

2.2 Our Neural Network Model

The figure for our neural network model is depicted in Figure 1. The block features for the nodes are input to shared layers of GNN to generate node embedding. To compute the node-level predictions, the node embedding is input to a feed-forward neural network (MLP). To compute the graph-level predictions, the same node embedding is input to a pooling followed by a separate MLP. Let ℓ_{graph} and ℓ_{node} denote the graph-level and node-level prediction loss, respectively. Then, the total loss is then weighted combination of ℓ_{graph} and ℓ_{node} where the weights w_{node} and w_{graph} are tuned to fix the imbalanced gradient norms using Grad-Norm [22], that is,

$$\ell_{total} = w_{node}\ell_{node} + w_{graph}\ell_{graph}. \tag{3}$$

This neural network structure allows the shared layers of GNN, which has many trainable parameters, to be trained through both gradients from node-level predictions and graph-level predictions.

3 Experiments

We conduct the following experiments to evaluate the effectiveness and efficiency of our framework.

Datasets We compare our method with several other hierarchical and flat methods with a several public datasets including Proteins, Proteins-Full (Proteins with node attributes included), and Enzymes [23, 24]. For all datasets, we evaluate model performance and report the mean accuracy over cross validation experiments. The details of the datasets are given in Appendix A Table 1

Setup We implemented our method using the Pytorch-Geometric framework [25] and evaluated our model with 10-fold cross validation. For each iteration, 10 percent of the training data was used

Method	Data Set		
	ENZYMES	PROTEINS	Proteins_full
GRAPHCONV [4]	94.13	**70.01**	86.24
CHEBCONV [17]	94.36	67.43	86.73
GAT [13]	79.82	66.38	73.24
SAGE [2]	76.84	62.4	71.33
OUR WORK	**95.41**	69.87	89.32
OUR WORK-TIED	94.76	65.26	**92.43**

(a) Predictive performances on node-level classification

Method	Data Set		
	ENZYMES	PROTEINS	Proteins_full
SORTPOOL	57.12*	75.54*	72.36
DIFFPOOL	62.53*	76.25*	74.39
SAGPOOL	62.26	71.86*	72.24
OUR WORK	**68.14**	**78.24**	**76.39**
OUR WORK-TIED	63.72	69.43	71.12

(b) Predictive performance on graph-level classification

Model	Enzymes	
	node	graph
Baseline	95.41	68.14
Uniform grad weights	94.36	61.13
Global pooling	96.71	52.67
Graph-head only	–	63.47
Cell-head only	96.74	–

(c) Ablative results. Global pooling implemented with global attention.

Figure 2: 10-fold cross-validation classification accuracies. * denotes statistic from [18, 19]

for validation. We optimized the parameters of our neural network with Adam [26] with learning rate 0.00025. We integrated dropout, ℓ-2 weight regularization, and an early stopping criterion with patience - we stopped training if the validation loss did not improve for 50 epochs with a maximum of 10k epochs following [27]. We adopt a grid-based hyperparamter selection scheme. The optimal parameters and details of our architecture are given in Appendix A Table 2. We use the ReLU activation function. Following [19], we compose representations from different scales by concatenating their pooled representations. For this purpose, we adopt a concatenated max and mean-pooling. Notably, the most significant parameter influencing the performance of our algorithm was the batch-size. We empirically found larger batch-sizes often result in overfitting on all graph datasets. A batch-size of 16 was used for our experiments. The tied method denotes additional regularization on the ℓ-2 difference between task prediction heads as opposed to independent prediction heads. In our experiment, we applied the regularization to all prediction heads, although for certain applications, it may be desirable to identify a subset of tasks to couple.

Results Our comparative 10-fold cross validation results are highlighted in Figure 2. We observe our framework's capability to avoid overfitting - particularly on the Enzymes dataset - and ability to improve performance even after reaching a solution which achieves near-perfect graph and node-level accuracy on the training set. We attribute these gains to our framework ability to learn multi-scale representations from hierarchical (node-level and graph-level) losses. Additionally, we provide the results of ablative experiments given in Table 2c. This table shows (i.) using tuned weights to fix the imbalanced gradient norms and our pooling layer significantly improves the graph-level predictions, (ii.) multi-task framework improves the graph-level prediction accuracy but slightly worsens the node-level prediction accuracy. This motivates us to explore methods for fine-tuning model weights for particular tasks post convergence on the multitask loss.

4 Conclusions and Future Work

We introduced a multi-scale graph prediction framework which can leverage error signals from multiple levels of the scale-hierarchy to learn effective and general representations through multi-task learning. We achieved strong performance compared to models which are exclusively applied to either graph-level or node-level predictive tasks and are also confident that additional performance gains can be achieved through more tuning. Our algorithm also shows robustness to overfitting. Future work includes exploring methods to improve stability and convergence during training without sacrificing test-set performance, integrating semi-supervised subgraph labels, coupling prediction nodes between tasks, evaluating additional convolution and pooling operators in our framework (e.g. Edge Contraction Pooling [20]) and studying the influence of synthetic labels or synthetic tasks as weak supervision.

References

[1] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *CoRR*, abs/1509.09292, 2015.

[2] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[3] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. *CoRR*, abs/1011.4071, 2010.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[5] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[6] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016.

[7] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Andrés Arbeláez, and Luc Van Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *CoRR*, abs/1701.04658, 2017.

[8] S. Xie and Z. Tu. Holistically-nested edge detection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, Dec 2015.

[9] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. Hierarchical multi-label classification networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5075–5084, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[10] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics (Oxford, England)*, 9:432–41, 08 2008.

[11] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

[12] Victor Sanh, Thomas Wolf, and Sebastian Ruder. A hierarchical multi-task approach for learning embeddings from semantic tasks. *CoRR*, abs/1811.06031, 2018.

[13] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2017.

[14] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, April 2016.

[15] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017.

[16] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017.

[17] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[18] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *CoRR*, abs/1806.08804, 2018.

[19] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. *CoRR*, abs/1904.08082, 2019.

[20] Frederik Diehl. Edge contraction pooling for graph neural networks. *CoRR*, abs/1905.10990, 2019.

[21] Enxhell Luzhnica, Ben Day, and Pietro Liò. On graph classification networks, datasets and baselines. *CoRR*, abs/1905.04682, 2019.

[22] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017.

[23] Aasa Feragen, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten Borgwardt. Scalable kernels for graphs with continuous attributes. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 216–224. Curran Associates, Inc., 2013.

[24] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(1):47–56, January 2005.

[25] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[27] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018.

Appendix A.

Data set	# Graphs	# Classes (graph)	# Classes (node)	# node attributes	Avg. # of Nodes per Graph	Avg. # of Edges per Graph
PROTEINS	1113	2	3	1	39.06	72.82
PROTEINS$_{full}$	1113	2	3	29	39.06	72.82
ENZYMES	600	6	3	18	32.63	62.14

Table 1: Details of data sets.

Module	Architecture
Encoder	GCN(128)-GCN(128)-MLP(128)
Graph-head	[GCN(128)-Pool(128, 0.5)]*3-MLP(256)
Node-head	GCN(128)
Prediction-head	MLP(128)-dropout(0.5)-MLP(64)-Lin

Table 2: Architecture parameters