CS224W Project: The Study of Drug-Drug Interaction Learning Through Various Graph Learning Methods

Olivia Hsu
Department of Computer Science
Stanford University
owhsu@stanford.edu

Chuanqi Chen
Department of Computer Science
Stanford University
cchuanqi@stanford

Abstract

Harm-causing Drug–drug interactions constitute an important concern in drug development and postmarketing pharmacovigilance since they expose patients to higher risks and increase public health system costs. Methods to follow-up and discover possible DDIs are a primary aim of drug safety researchers. These drug-drug interactions can be represented as an edge prediction task on an undirected, unweighted graph, which can be found in the Open Graph Benchmark (OGB) in the ogbl-ddi dataset. In this work, we implement three recently published graph machine learning model frameworks to predict existing, and potentially new, drug-based interactions on the ogbl-ddi dataset. The models we implemented are 1) Adaptive Graph Encoder [Cui et al. (2020)]), 2) DeeperGCN [Li et al. (2020)], and 3) Low Rank Graph Attention (LRGA) for Graph Convolutional Networks [Puny et al. (2020)]. For the baseline models that performed relatively well on ranking unseen drug interaction links, we also performed certain enhancements, like feature augmentation and a parameter ablation study, to improve their performance. This study aims to understand, motivate, and improve-upon existing graph learning models for the purpose of applying them to drug-drug interaction applications specifically.

1 Introduction

Drug–drug interactions (DDIs) are a serious problem in patient safety [Percha and Altman (2013), Becker et al. (2007)]. Co-administration of two or more drugs at the same time can affect the biological action of the implicated drugs. The main types of DDIs include pharmacokinetic and pharmacodynamic interactions [Aronson (2004)]. Pharmacokinetic interactions can affect important drug processes that determine bioavailability, such as absorption, distribution, metabolism and excretion [Aronson (2004)]. Examples of these interactions are the administration of a medication that increases the motility of the intestine decreasing the absorption of the other drug, competition for the same plasma protein transporter, inhibition of the action of a metabolizing enzyme or even interaction at excretion level affecting the elimination of one of the drugs [Palleria et al. (2013)]. On the other hand, pharmacodynamic interactions can occur at the pharmacological receptor level with both drugs interacting with the same protein, at the signaling level affecting different signaling pathways, or at the effector levels causing different pharmacological responses. DDIs result in many adverse drug effects (ADEs) that can cause severe injuries to patients and even be responsible for deaths [Lazarou et al. (1998)]. It has been reported that DDIs could be responsible for up to 30% of the adverse effects found in the patients [Pirmohamed and Orme (1998)]. Hospitalizations and emergency department visits because of coadministration of different drugs are estimated around 0.57 and 0.054%, respectively [Becker et al. (2007)].

1.1 Dataset

We conducted link prediction on the Open Graph Benchmark (OGB) Drug-Drug Interation (ogbl-ddi) dataset. The ogbl-ddi dataset is an unweighted, undirected graph, where each node represents an FDA-approved or experimental drug from the DrugBank 5.0 database and each edge represents interactions between the corresponding drugs [Wishart et al. (2018)]. The interaction can be interpreted as a phenomenon where the joint effect of taking two drugs together is considerably different from the expected behavior of taking each drug independently. The ogbl-ddi link prediction task aims to rank true positive drug interactions above approximately 100,000 randomly-sampled negative drug interaction edges using the Hits@20 metric. The statistics of the dataset are shown in Table 1

Table 1: Dataset Statistics

Dataset	# Nodes	# Edges	# Features	# Classes
ogbl-ddi	4,267	1,334,889	0	0

2 Existing Method and Our Method

2.1 Hypothesis

Since the ogbl-ddi does not have any node level or edge level attributes or adjacency weights, the link prediction task is based solely on the undirected graph structure. To further learn the link and connectivity structure of the graph, we can preprocess the graph to capture node-level properties, such as the clustering coefficient or generalized degree of each node. Additionally, adding these node-level properties should increase the structural information represented in the graph framework through feature augmentation. In addition to feature augmentation, utilizing node embeddings that capture neighborhood structure through random walks can also increase the predictive power of the graph learning models.

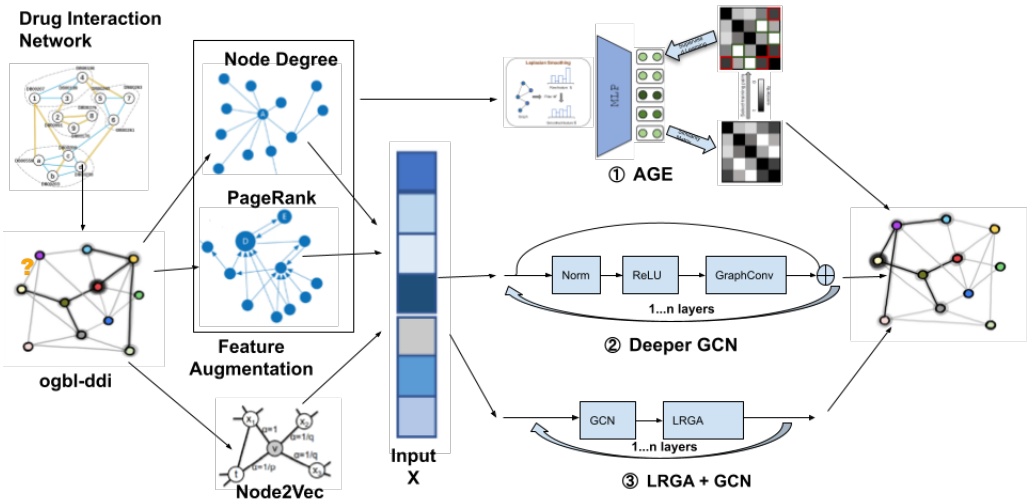

Figure 1: DDI Graph Networks

2.2 Adaptive Graph Encoder

2.2.1 Motivation

Recently, Graph Convolutional Networks have made progress towards the task of attributed graph embeddings, the ability to learn node embedding vector representations using both node features and

graph topology combined. However, existing GCN methods have three main limitations: (1) The intertwined combination of filters and weight matrices in GCNs negatively affect both the model's performance and robustness, (2) graph convolutional filters in GCNs do not preserve optimal low-pass filtering characteristics even though they are a special example of Laplacian smoothing filters, and (3) GCNs usually recover the adjacency matrix or feature matrix of a graph, which is not always the best training objective for certain applications.

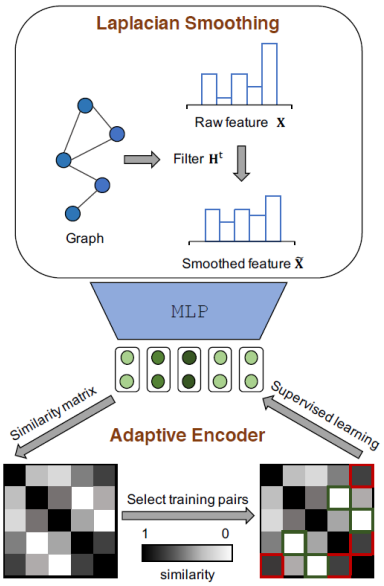

Figure 2: Adaptive Graph Encoder[Cui et al. (2020)]

In order to resolve the above issues with GCN learning techniques, Cui et al. (2020) proposes an Adaptive Graph Encoder (AGE) [Figure 2], which trains on node embeddings that have been transformed by a Laplacian smoothing filter to remove high-frequency noise and uses a dynamic node-pair sampling strategy to train an adaptive encoder. Experimental results demonstrate that their model outperforms state-of-the-art graph embedding methods on four networking tasks, therefore, we believed that it would also be promising when applied to the ogbl-ddi dataset. A traditional GCN is a form of the Laplacian smoothing filter where $k = 1$, but in this work the authors improve upon the model by choosing the optimal $k = 1/\lambda_{max}$. Through an ablation study, they found that $k = 1/\lambda_{max}$ maximizes the low-pass region of the filter while still denoising the high-frequency components of the node attribute vector, where $\lambda_{max} = 3/2$ is approximately equal to largest eigenvalue of the Laplacian for 5 datasets. Next, the paper argues that recovering the similarity matrix is a more accurate optimization objective than recovering the adjacency matrix or node features because similarity matrix recovery captures both node attribute and local graph structure information. The AGE algorithm does this by choosing positive and negative samples by thresholding similarity matrix scores using r_{pos} and r_{neg}, respectively. The encoder is then trained using the cross-entropy loss on the chosen node-pair samples and their corresponding similarity scores. And finally, the r_{pos} and r_{neg} thresholds are updated at regular intervals throughout training to allow for refinement of the training sample region. Cui et al. (2020) showed their results on a link prediction task using the Cora dataset with parameters: $t = 8$, $r^{st}_{pos}/n^2 = 0.0110$, $r^{st}_{neg}/n^2 = 0.1$, $r^{ed}_{pos}/n^2 = 0.0010$, and $r^{ed}_{neg}/n^2 = 0.5$.

2.2.2 Limitations

Although we originally believed that the AGE model would work for the ogbl-ddi dataset since it performed well on the Cora dataset for link prediction, we ran into many limitations for this graph encoding model. One of the main issues is that the ogbl-ddi dataset is inherently attribute-free, which meant that the task of attributed graph embeddings was significantly harder to solve. Since the ogbl-ddi graph did not have any, we tried using feature augmentation to create graph embeddings.

However, this only created about 5-10 node features, which was not enough features to embed into a 500 dimensional embedding space. Another limitation of the AGE model is its scalability to larger datasets. The AGE baseline model in Cui et al. (2020) learns the similarity matrix on the Cora dataset, which only has 2,708 nodes, 5,429 edges, and 1,433 features. However, materializing the dense similarity matrix for the ogbl-ddi dataset resulted in implementation limitations on the hardware. The last problem we encountered with implementing the AGE model using the ogbl-ddi dataset involved setting the adaptive threshold values for r_{pos} and r_{neg}. Although adaptive thresholdign can be beneficial, as stated above and by the authors, it can also be harmful if the thresholds are not properly set and the graph embedding framework does not have enough samples to train and learn from. All of this led to the AGE model performing extremely poorly on the ogbl-ddi dataset (with an average training Hits@20 metric of about 0.00) and caused us to pivot towards another method instead.

2.3 Deeper Generalized GCN

2.3.1 Motivation

As previously mentioned, GCNs have made significant progress towards the task of learning on graphs. Current GCNs tend to be shallow and usually contain no more than four layers [Zhou et al. (2019)]. Unlike traditional Convolutional Neural Networks (CNNs), stacking increasingly deep layers in GCNs do not improve performance because they provably suffer from oversmoothing and vanishing gradients. Increased GCN layers also worsen the problem of model overfitting. Therefore, Li et al. (2020) proposes a graph learning framework called DeeperGCN that enables deeper GCN layers. DeeperGCN defines a generalized aggregation network's message passing function as:

$$\mathbf{m}_{vu}^{(l)} = \text{ReLU}(\mathbf{h}_u^{(l)} + \mathbb{1}(\mathbf{h}_{e_{vu}}^{(l)}) \cdot \mathbf{h}_{e_{vu}}^{(l)}) + \epsilon, u \in \mathcal{N}(v) \tag{1}$$

where the generalized message aggregation function is permutation invariant to message ordering. DeeperGCN also enables deep learning by creating a novel DeeperGCN layer architecture that modifies previous literature defining skip/residual connections and defines a new message normalization layer. The previously defined residual connection in DeepGCN [Zhou et al. (2019)] applies skip connections after the architecture Conv –> Norm –> Act –> Addition, whereas DeeperGCN [Li et al. (2020)] redefines a pre-activation connection as Norm –> Act –> Conv –> Addition for each combined model layer in the framework.

The DeeperGCN Model seemed like a good fit for the ogbl-ddi dataset since Li et al. (2020) showed DeeperGCN's scalability to various large-scale graphs in the OGB dataset suite (ogbn-proteins, ogbn-arxiv, ogbg-ppa, ogbg-molhiv). Since the ogbl-ddi dataset also does not have any node or edge-level attributes, multi-layer GCNs should be better at recovering deeper large-scale graphs structures as long as the oversmoothing and overfitting issues are addressed. The DeeperGCN framework for the DDI application seemed encouraging because the work argued that their generalized convolutional architecture resolved overfitting, oversmoothing, and vanishing gradients. We were also motivated to apply this model to the ogbl-ddi dataset over the previous AGE model because baseline GCN methods achieved extremely high Hits@20 metrics for this application. Message passign methods and simpler graph convolution networks, like GraphSage, already performed relatively well at learning the edge structure of the graph.

2.3.2 Limitations

Various implementation and algorithmic limitations arose when applying the DeeperGCN method to the ogbl-ddi dataset. Due to the large number of network layers and parameters in the DeeperGCN method, physical hardware would often run out of memory with any hidden channel size approximately larger than 64. Physical memory limitations result in a tradeoff between deeper networks versus larger networks for large graph learning problems. For the ogbl-ddi dataset, we found that wider networks were more expressive in representing the edge structure of the graph because the higher dimensionality preserves enough high-frequency graph structure information. Another limitation of deeper GCN architectures is the amount of time it takes to train the given model. Increased model depth leads to an exploding number of parameters and increases the gradient backpropagation computation, leading to a runtime that is over an order of magnitude slower than other models. DeeperGCN was also unable to curb overfitting even with the modified skip connections, message normalization layers, and generalized convolution layers. Even though the paper claims that the

DeeperGCN framework enables more performant deep GNN networks, the method still does not train well on ogbl-ddi (see Table 2).

2.4 Low Rank Global Attention on Graph Convolutional Networks

2.4.1 Motivation

Similar to the DeeperGCN reasoning, the Low Rank Global Attention (LRGA) on GCN method was then chosen because of how the algorithm enhances and improves upon GCNs. Since the GraphSage and GCN baselines already performed well on the ogbl-ddi dataset, we looked at methods that further enhanced the link prediction results of graph neural network (GNN) related methods. Attention mechanisms, first developed in the context of Natural Language Processing [Bahdanau et al. (2016)], has proven to also be a powerful model in graph neural networks. Attention can be intuitively thought of as creating an adaptive importance metric between pairs of inputs, specifically node features in the graph learning concept, where global attention creates that metric globally between all pair-wise combinations of inputs. Puny et al. (2020) propose a low-rank (lower dimensional) MLP-based global attention transformation network that is applied after each GNN layer. The MLP-based global attention module globally aggregates all inputs $\mathbf{X}$ into low-dimensional space, κ, via following update rules, where m_1, m_2, m_3, m_4 are MLPs operating on the feature.

$$X^{l+1} \leftarrow [X^l, \text{LRGA}(X^l), \text{GNN}(X^l)] \tag{2}$$

$$\text{LRGA}(X) = \left[\frac{1}{\eta(X)} m_1(X)(m_2(X)^T m_3(X)), m_4(X)\right] \tag{3}$$

$$\eta(X) = \frac{1}{n}(\mathbf{1}^T m_1(X))(m_2(X)^T \mathbf{1}) \tag{4}$$

The paper also proves that Random Graph Neural Networks (RGNNs) with LRGA are as expressive as 2-Folklore Weisfeiler-Lehman (2-FWL) algorithm , making them as more expressive than vertex coloring which is the provable limit of message-passing GNNs. Our reasoning behind using LRGA for the ogbl-ddi domain was that it could also improve upon multiple GNN architectures, like GraphSage and GCNs, without much computational overhead which was one of our previous issues with DeeperGCN's architecture.

2.4.2 Further Improvements

We applied LRGA to various GNNs including GCNs, GraphSage, and DeeperGCN's Generalized Convolutional Aggregation method. Additionally, LRGA was proven in Puny et al. (2020) to be expressive using RGNNs, but the randomness of RGNNs do not include graph information in the input node embeddings. Therefore, to further improve LRGA's performance, we decided use global attention on GCNs with Node2Vec embeddings instead of random ones. This also aligns with our hypothesis since Node2Vec input node embeddings should carry more graph structure information learned from the biased random walks [Grover and Leskovec (2016)]. The biased random walks allow the embeddings to capture both local and global link structures around each node through a combined BFS and DFS search. In addition to capturing neighborhood information from Node2Vec, we further passed more graph structure information to the LRGA augmented GCN method by concatenating the Node2Vec embeddings with feature augmentation. The feature properties we included for each node are: node degree, clustering coefficient, pagerank, betweenness centrality, square clustering. etc. This allows for the model to learn the link-structure of the ogbl-ddi dataset using not just random walk information but also information about steady state flow, neighborhood clustering, node connection counts, and more.

3 Experiments

3.1 Method Comparison

3.1.1 Baselines

We compare our performance with the following state of the art baselines: MAD Learning [Luo et al. (2021), GCN [Kipf and Welling (2017)], GraphSage [Hamilton et al. (2018)], and the original LRGA

+ GCN method from [Puny et al. (2020)]. All baseline code, metrics, and papers can be found on the Open Graph Benchmark ogbl-ddi leaderboard.

3.1.2 Evaluation Protocol

Test results for all methods are reported by the best validation epoch averaged over 10 random seed runs.

3.1.3 Implementation Details

All Methods. The embedding dimension split for all methods that used node embeddings was calculated as:

$$\text{Node2Vec dim} = n_{\text{Node2Vec}} = \text{floor}(\frac{1}{n_{\text{embedding types}}} * (n_{\text{hidden channels}} - n_{\text{augmented features}})) \quad (5)$$

$$\text{Random dim} = n_{\text{random}} = n_{\text{hidden channels}} - n_{\text{Node2Vec}} - n_{\text{augmented features}} \quad (6)$$

As an example, the LRGA + GCN (random + Node2Vec + aug) had 5 features and 2 embedding types (random and Node2Vec) with a hidden channel size of 512. All methods that included feature augmentation had the following node properties added: pagerank, clustering coefficient, generalized degree, node centrality, and a constant value of 1, with $n_{\text{augmented features}} = 5$. For all the non-baseline ablated models in Table 2 an Adam optimizer with a learning rate of 0.005 was run for 200 epochs. We also ran all non-baseline models in Table 2 with a batch size of $64 * 1024 = 65536$ and a dropout rate of 0.5.

DeeperGCN Only. The DeeperGCN model was run with 2 generalized combined layers each with 2 MLP layers. Each generalized combined layer had a ReLU activation with a layer normalization [Ba et al. (2016)] followed by a generalized convolution layer with a hidden channel size of 64. The generalized convolution layer was implemented with the SoftMax aggregation function.

LRGA Methods Only. $\kappa = 50$ for all LRGA variant methods, including the baseline. The LRGA module are implemented according to equations 2, 3, and 4. Each LRGA module contains 4 MLPs m_1, m_2, m_3, m_4. Each m_i is a single linear layer with ReLU activation, batch and graph normalization are used at each layer. The only difference of hyper parameters among these five methods in Table 2 are hidden channels: LRGA + GraphSage (Node2Vec): 512, LRGA + GraphSage (Node2Vec): 512, LRGA + GCN (random + aug): 512, LRGA + GCN (Node2Vec + aug): 1024, LRGA + GCN (random + Node2Vec + aug): 512, LRGA + GCN + SkipConnect (Node2Vec + aug): 512.

3.1.4 Benchmark Results

Table 2 summarizes the results of training and evaluating our model according to the evaluation protocol against baseline implementations. We observe that the LRGA + GCN with Node2Vec embeddings and feature augmentation (denoted in Table 2 as 'aug') performs the best, supporting our claim that global attention, graph neural networks, random walk embeddings, and feature augmentation combined do a better job of capturing graph structure information. All of our LRGA input modifications perform better than the original result published in [Puny et al. (2020)], which emphasizes the importance of choosing a representative input into the graph network. Our method achieves state-of-the-art performance on the ogbl-ddi dataset.

Table 2: Benchmark Comparison Results

Method (embedding)	**Hits@20(%)** Training	Validation	**Test**
MAD Learning		70.10 ± 0.82	67.81 ± 2.94
LRGA + GCN (random)		66.75 ± 0.58	62.30 ± 9.12
GraphSAGE		53.90 ± 4.74	62.62 ± 0.37
GCN		37.07 ± 5.07	55.50 ± 2.08
DeeperGCN (random + aug)	64.71 ± 1.49	56.92 ± 1.33	31.52 ± 8.27
LRGA + GraphSage (Node2Vec)	77.86 ± 1.02	68.27 ± 0.96	61.23 ± 13.62
LRGA + GCN (random + aug)	79.73 ± 0.61	69.85 ± 0.60	66.55 ± 8.70
LRGA + GCN (Node2Vec + aug)	**82.93 ± 0.55**	**72.25 ± 0.47**	**73.85 ± 8.71**
LRGA + GCN (random + Node2Vec + aug)	79.64 ± 0.86	69.58 ± 0.90	70.66 ± 5.88
LRGA + GCN + SkipConnect (Node2Vec + aug)	81.11 ± 0.93	71.66 ± 1.38	65.91 ± 11.22

3.2 Parameter Ablation Study

After determining that low rank graph attention on graph convolutional networks with Node2Vec input embeddings and feature augmentation – denoted as LRGA + GCN (Node2Vec + aug) – was the best method, we further conducted a parameter ablation study on this method. We investigated the affects of certain parameters on the performance of the GCNs augmented with LRGA, Node2Vec and feature augmentation. Our experimental setting included fixing the models baseline parameters to mirror that of the original LRGA paper, which used parameters denoted by the asterisk (*) in Table 3.

Table 3: Parameter Ablation Study for Model **LRGA + GCN (Node2Vec + augment)**

Hits@20(%) Validation	**Test**	Hidden Channels	Learning rate	Num layers	Batch size	Dropout
67.28 ± 0.74	60.80 ± 10.45	512	**0.002**	2	65536	0.5
70.13 ± 0.50*	73.41 ± 7.15	512	**0.005**	2	65536	0.5
71.33 ± 0.52	67.96 ± 10.41	512	0.005	2	65536	**0.2**
70.13 ± 0.50*	73.41 ± 7.15	512	0.005	2	65536	**0.5**
70.13 ± 0.50*	73.41 ± 7.15	512	0.005	2	**65536**	0.5
67.28 ± 2.10	62.69 ± 5.65	512	0.005	2	**100000**	0.5
70.62 ± 1.01	72.91 ± 5.07	512	0.005	2	65536	**0.45**
70.13 ± 0.50*	73.41 ± 7.15	512	0.005	2	65536	**0.5**
67.52 ± 0.75	50.33 ± 13.37	512	0.005	**1**	65536	0.5
70.13 ± 0.50*	73.41 ± 7.15	512	0.005	**2**	65536	0.5
65.32 ± 5.56	64.48 ± 14.18	512	0.005	**3**	65536	0.5
65.52 ± 0.87	56.22 ± 10.15	**256**	0.005	2	65536	0.5
70.13 ± 0.50*	73.41 ± 7.15	**512**	0.005	2	65536	0.5
70.55 ± 0.31	73.51 ± 8.69	**640**	0.005	2	65536	0.5
71.54 ± 0.61	74.24 ± 14.18	**768**	0.005	2	65536	0.5
71.73 ± 0.65	73.13 ± 13.79	**800**	0.005	2	65536	0.5
71.81 ± 0.73	**77.94 ± 9.20**	**832**	0.005	2	65536	0.5
72.03 ± 0.59	74.17 ± 13.97	**896**	0.005	2	65536	0.5
71.65 ± 0.53	75.88 ± 10.28	**960**	0.005	2	65536	0.5
72.25 ± 0.47	**73.85 ± 8.71**	**1024**	**0.005**	**2**	**65536**	**0.5**
70.97 ± 2.76	66.07 ± 20.60	**1152**	0.005	2	65536	0.5

Even though the ablation study swept through many of the parameters, we observed that the hidden channel dimension had the biggest impact on our Node2Vec embedding method. Since the number of hidden channels is directly proportional to the Node2Vec embedding dimension (see Equation 5), the dimensionality directly relates to the information obtained from the Node2Vec random walk

algorithm. Intuitively, higher dimensionality can represent more high-frequency graph structures while lower dimensionality results in graph smoothing affects. Although increasing the number of hidden channels tends to improve expressivity, too many hidden channels will not allow the graph learning framework to remove high frequency noise and learn lower-dimensional properties of the graph. The Node2Vec embedding space needs to be large enough to differentiate between node similarities and differences for the 4,267 nodes, which is $4267 * 4267 = 18,207,289$ node pairs, in the ogbl-ddi dataset. Unlike the original LRGA + GCN model in Puny et al. (2020), Node2Vec embeddings capture more local and global graph structure information than random embeddings so changes in the embedding space dimensionality should affect our model more. Based on the ablation study, we found that the optimal number of hidden channels was 1024.

4 Conclusion

To push state-of-the-art drug-drug interaction prediction, we hypothesized the importance of node properties, augmented node features, and node embeddings for link prediction learning on an attribute-less network. We explained the motivation behind this hypothesis and how this led us to look at several state-of-the-art models: AGE, DeeperGCN, and LRGA + GCN. We then demonstrated how these various state-of-the-art graph learning methods still have limitations and fall short of capturing this node information. Empirically, we also show the effectiveness of choosing the node information for frameworks to learn on through both a model comparison and a parameter ablation study. This all lead to setting a new state-of-the-art on the ogbl-ddi dataset in the Open Graph Benchmark.

5 Acknowledgements

We would like to acknowledge Puny et al. (2020), Cui et al. (2020)]), Li et al. (2020), and Fey and Lenssen (2019) for the use of various parts of their code in our implementation.

References

Cui, G.; Zhou, J.; Yang, C.; Liu, Z. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* **2020**,

Li, G.; Xiong, C.; Thabet, A.; Ghanem, B. DeeperGCN: All You Need to Train Deeper GCNs. 2020.

Puny, O.; Ben-Hamu, H.; Lipman, Y. Global Attention Improves Graph Networks Generalization. 2020.

Percha, B.; Altman, R. *Trends in pharmacological sciences* **2013**, *34*, 178–84.

Becker, M.; Kallewaard, M.; Caspers, P.; Visser, L.; Leufkens, H.; Stricker, B. *Pharmacoepidemiology and Drug Safety* **2007**, *16*.

Aronson, J. *British journal of clinical pharmacology* **2004**, *58*, 343–4.

Palleria, C.; Paolo, A. D.; Giofrè, C.; Caglioti, C.; Leuzzi, G.; Siniscalchi, A.; Sarro, G. D.; Gallelli, L. *Journal of Research in Medical Sciences : The Official Journal of Isfahan University of Medical Sciences* **2013**, *18*, 601 – 610.

Lazarou, J.; Pomeranz, B.; Corey, P. *JAMA* **1998**, *279*, 1200—1205.

Pirmohamed, M.; Orme, M. *Davies's textbook of adverse drug reactions* **1998**, 888–912.

Wishart, D. et al. *Nucleic acids research* **2018**, *46*.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph Neural Networks: A Review of Methods and Applications. 2019.

Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2016.

Grover, A.; Leskovec, J. node2vec: Scalable Feature Learning for Networks. 2016.

Luo, Y.; Chen, A.; Hui, B.; Yan, K. Memory-Associated Differential Learning. 2021.

Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. 2017.

Hamilton, W. L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. 2018.

Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer Normalization. 2016.

Fey, M.; Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric. ICLR Workshop on Representation Learning on Graphs and Manifolds. 2019.