Comparison between Inductive and Transductive Learning in a Real Citation Network using Graph Neural Networks

Guillaume Lachaud*, Patricia Conde-Cespedes†, Maria Trocan‡
ISEP - Institut Supérieur d'Électronique de Paris
10 rue de Vanves, Issy-les-Moulineaux, 92130-France
Email: *glachaud@isep.fr, †pconde@isep.fr, ‡maria.trocan@isep.fr

***Abstract*—Graph data is present everywhere and has vast ranging applications from finding the common interests of people to the optimization of road traffic. Due to the interconnectedness of nodes in graphs, training neural networks on graphs can be done in two settings: in transductive learning, the model can have access to the test features in the training phase; in the inductive setting, the test data remains unseen. We explore the differences between inductive and transductive learning on real citation networks when the graphs are converted to undirected graphs. We find that the models achieve better accuracy in the transductive setting than in the inductive setting, but that the gap between validation and test accuracy is also higher, which indicates the models trained in an inductive setting have better generalization capabilities.**

I. Introduction

Graphs are present in almost all aspects of society and in many fields of scientific studies. We spend a fair amount of time every day surrounded by graphs: when we interact with other people on social networks, browse content on streaming platforms, buy products using online retailers, finding out the best itineraries for travelling from one place to another. In science, epidemiologists can observe the spread of a disease; programming code can be analyzed using its abstract syntax tree.

All this manipulation of graph structured data requires efficient methods. Following the successes of deep learning methods in computer vision [1] and natural language processing [2], there has been significant development in the area of graph representation learning using deep learning, with the use of Graph Neural Networks (GNNs)[3]. Many concepts from Convolutional Neural Networks (CNNs)[4] and Recurrent Neural Networks (RNNs)[5] have efficiently been transposed to GNNs[6], [7].

Due to the interconnectedness of nodes in graph data, it is not possible to create a series of independent examples to be fed to a neural network, as can be done in computer vision with images or in natural language processing with sequences of text input. Thus, there are two approaches for splitting a dataset between training and test data: the unlabeled features of the test nodes can be used during the training phase of the model, or they can be ignored. The first approach corresponds to the *transductive* setting, while the latter represents *inductive* learning [8]. In transductive learning, the model tries to predict labels of nodes already seen in the training phase, while the model in inductive learning is tested on unseen nodes. Inductive learning is therefore better suited to improve the generalization power of a model.

IEEE/ACM ASONAM 2022, November 10-13, 2022
978-1-6654-5661-6/22/$31.00

In this paper, we investigate the training of GNNs in transductive and inductive settings, where the dataset is a citation network which can change over time. We evaluate how much validation and test data is used in training in the transductive setting. We also present baselines of state-of-the art architectures in the inductive setting on the citation of network of computer science papers published on arXiv.

The paper is organized as follows. Section II presents the main concepts, namely a brief overview of GNNs, and in particular the GNNs architectures that we use in the experiments. Section III explains the problem of dataset splitting for graphs. Section IV describes the *ogbn-arxiv* dataset, and provides statistics about *ogbn-mag* and *ogbn-papers100M* in terms of the distribution of the edges in the data splits. Section V introduces and discusses the results. Section VI concludes the work.

II. Main definitions

Some of the more important advances in the field of GNNs include the formalization of the generic architecture of a GNN and the exploration of various neighbourhood selection schemes. Furthermore, the development of new architectures has mostly relied on adapting successful techniques from other domains, such as attention mechanisms and knowledge distillation, or on exploiting unique properties of graph stuctured data, such as reversible GNNs or GNNs that exploit information from multihop neighbours. Another particularity of GNNs is the possibility to leverage the graph information in the design of the features. Besides, the dependencies between nodes play a role in how a graph is split between

training and test data, which can affect model performance. In the rest of this section we present the main concepts that will be used throughout the paper.

Message Passing Neural Framework

Most GNNs follow the message passing neural network framework introduced in [9]. The network aggregates information about the neighbors of a node to produce messages that are used to update the hidden representation of each node. More formally, let $\mathcal{G} = \mathcal{V}, \mathcal{E}$ be a graph where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges. h_v^l represents the features of a node v at the layer l, and H^l the feature matrix for all the nodes. e_{vw} represents the edge from node v to node w. If X denotes the node features matrix, we have by convention $H^0 = X$. An update from layer l to layer $l+1$ takes the form

$$m_v^{l+1} = \sum_{w \in \mathcal{N}_v} M_l(h_v^l, h_w^l, e_{vw}) \quad (1)$$

$$h_v^{l+1} = U_l(h_v^l, m_v^{l+1}) \quad (2)$$

where M_l and U_l are respectively the message and update functions of layer l. The authors in [10] showed that theoretically optimal GNNs must be permutation invariant with respect to the nodes, i.e. the order in which the nodes are input does not change the result of the network.

Neighborhood selection

The choice of neighbourhoods for each node can affect the performance of the network. Classical graph neural networks such as Graph Convolutional Networks [3] used the full neighborhoods of each node. To be able to handle larger graphs, some approaches apply neighbor sampling such as GraphSAGE [11], while trying to preserve the graph structure as in Cluster-GCN [12].

Transductive and inductive learning

The concepts of transductive and inductive learning are related to the choice of neighborhood. In the transductive setting, the neighbours of the training nodes can belong to the validation and test sets; in this case, only their features are known at training time. In the inductive setting, the neighbours of the training nodes are restricted to other training nodes. The nature of the dataset, in particular in the way it evolves over time, can influence the preferred mode of learning, i.e. when it is better to train the model using transductive learning, and when it is better to use inductive learning instead [8]. It is especially important in temporal graphs [13], [14].

Attention mechanisms

Attention mechanisms were introduced in [5] in the context of natural language processing, with the goal of learning which parts of a sentence are the most important ones. They were successfully imported to GNNs with the introduction of the Graph Attention neTwork (GAT) in [7] and remain part of most of the leading architectures [15], [16], [17].

Self-knowledge distillation

Another approach used for decreasing memory load is knowledge distillation [18]. A large neural network is trained and acts as a teacher for training smaller networks, the students. Using this principle, it was shown in [4] that an efficient way to implement knowledge distillation is to train the early layers of a network as the students and the whole network as the teacher.

Reversible GNNs

GNNs are memory intensive, to the point that many models cannot be run with large datasets on standard GPUs. Most of the memory cost comes from having to store the features tensor of all the graph at each layer. To alleviate this problem, *reversible* graph neural networks have been proposed in [19], [6]. The idea is to propagate the information from layer to layer using a chain of operations that will be performed in an inverse way for the backpropagation. It increases the time complexity in order to decrease the space complexity.

Adaptive Graph Diffusion Networks (AGDN)

In addition to memory constraints, adding layers to GNNs can lead to oversmoothing of the features, where the information of a single node is drowned in the information coming from all the graph [20]. One way to counteract this effect is to try and retrieve information at each layer from different *hop-neighborhoods*; that is, nodes that are one-hop apart (direct neighbours), those that are 2-hop apart (neighbours of neighbours), and so on. One example of such architecture was proposed in [16].

Graph Information Aided Node feature exTraction (GIANT)

The choice of features associated with each node and edge can play a significant role in the efficiency of a classifier. Most methods of feature extraction using raw data, such as word2vec [21] or BERT [2] do not leverage the graph topology when constructing the new features. The Graph Information Aided Node feature exTraction (GIANT) framework was proposed in [15] to incorporate the graph topology in the feature extraction.

Dataset splitting

A naive approach to splitting a graph between train, validation and test sets, is to randomly assign each node to one group. This is what was done in [8] on graphs like CiteSeer, Cora and Pubmed [22]. Similarly, the authors of [12] follow a randomized split without validation set for training their model. By contrast, the authors of [23], who

incorporated the Amazon dataset in [12] as the *ogbn-products* in OGB, recommend using a split that relies on the sales ranking of the products: the most sold items are in the training set while the most rarely sold are in the test set. The authors further argue that this split matches the behaviour of real-world applications where, due to the cost of labelling data, the most important nodes are labelled first and a model is used to infer the labels of the less important nodes.

III. Problem description

In contrast to image and text data, graphs cannot easily be separated into multiple datasets, because nodes are interconnected with each other. In order to train a GNN, we must create a training set on which the model is trained; a validation set whose purpose is to find the best model; and a test set containing unseen data to see if the model is performing well.

When the graph can dynamically evolve over time, as is the case for social networks or citation networks, the splitting is usually based on the temporality of the nodes: the earliest nodes are put in the training data and the most recent nodes form the test data. This is coherent with the goal of most of the applications using temporal graphs: being able to predict future nodes features, edges creation, etc., using a model trained on the existing data.

In transductive learning, the nodes in the training data might have access to the features of the nodes in the validation and test sets. Consider the node in Figure 1. For simplification purposes, edges between other nodes than the central nodes are removed. The edges are represented as undirected, and the self-loop has been added. Here the node has 48 neighbours (not counting itself) with the following distribution: 13 nodes are in the training set, 16 in the validation data and 19 in the test set. Each additional training node might contain other edges with nodes in the validation data and test data. After several layers of a GNN, and depending on the locality of the node in the graph, the node has received information about many of the nodes in the validation and test sets. In the inductive setting, the model is trained using only the subset of nodes in the training data; in our example, this represents the blue nodes. Since most of the GNN entries on the Open Graph Benchmark leaderboard [1] train their models in a transductive setting, there is a spread of information from the validation and the test data to the training data.

In conjonction with exploiting the features of nodes in the test set, most methods use label information as a way to augment the features of the nodes: given a set of features X_{feats}, a one hot encoding vector is used to represent the label of the node. The new features X_{onehot} are then concatenated with X_{feats} to produce the input features X. For the elements outside of the training set, the label is first left empty; i.e. all the entries in the one hot vector are set to zero. Then the vector is filled by applying a softmax to the output logits of the GNN. If there is information from the test set that is contained in the validation data, it means that the network will indirectly try to produce the best label features for the test nodes that are used in classifying the validation nodes. It is thus trying to fit the test data.

[1]The leaderboard can be accessed at https://ogb.stanford.edu/docs/leader_nodeprop/.

IV. Datasets

Some of the most used dataset benchmarks come from the Open Graph Benchmark (OGB) [23], which contains datasets for node prediction, link prediction and graph prediction tasks. OGB has homogeneous and heterogeneous graphs of varying sizes, directed and undirected graphs from a variety of applications, such as protein-protein interactions, molecular graphs, knowledge graphs, products graphs or citation networks. Additionally, OGB also offers large scale graph challenges, where the graphs contain millions of nodes. [24]. In this paper we focus on the *ogbn-arxiv*, *ogbn-papers100M* and *ogbn-mag* datasets.

obgn-arxiv contains all the computer science papers in the arXiv repository that are indexed by the Microsoft Academic Graph (MAG) [25]. It is a directed graph with 169,343 nodes and 1,166,243 edges. Each node contains a 128-dimensional feature vector that represents the average embedding of the word embeddings in the title and the abstract, where the embedding function is a word2vec [21] that was trained on MAG [25]. Each node additionally contains a year attribute which indicates the year of publication.

The label of a node is the subject to which the paper belongs. Each class represents one of the 40 categories in the arXiv computer science repository. More exhaustive information about these classes can be found at https://arxiv.org/archive/cs.

ogbn-papers100M is a dataset constructed in a similar way to *ogbn-arxiv*, except that it contains more than 111 million papers and that the classification task is performed on the subset of papers that corresponds to all the papers published on the arXiv website; it is not restricted to computer science and there are 172 subject areas.

ogbn-mag is a heterogeneous graph which contains about 736,000 papers, 1.1 million papers, 8,700 institutions and 60,000 fields of studies. The features of the nodes are constructed with the same approach that was used for creating the features of *ogbn-arxiv*. The task is to predict the venue of the paper.

To ease manipulation with common GNNs, these graphs are usually converted to an undirected graph by adding all the

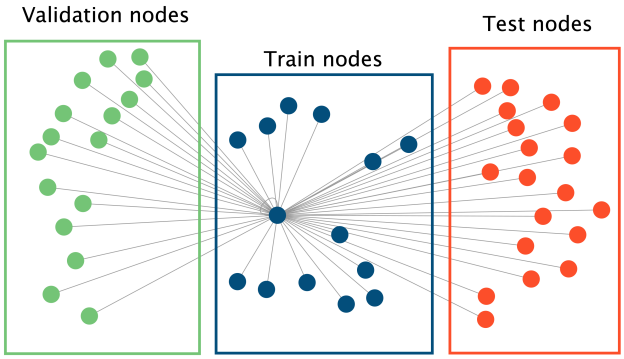

Fig. 1. Node in the training set with neighbors in all the graph.

reverse edges to the network [2], while also adding self-loops to each node. Transforming the graph into an undirected one makes the adjacency matrix symmetric, and thus easier to handle. Adding self-loops allows each node to pass its hidden representation to the next layer; without self-loops, the representation of the node at the next layer might not contain any information from the features of the node at the previous layer.

When training a GNN on a social citation networks from OGB, *ogbn-arxiv*, *ogbn-mag* and *ogbn-papers100M*, the recommended data split by the authors of [23] is to put all the papers published before 2018 in the training set; those published in 2018 in the validation set and the rest in the test set. Table I shows the distribution of nodes and edges in *ogbn-arxiv*, *ogbn-mag* and *ogbn-papers100M*, as well as the number of edges that have a source and a target node in a different part of the data split, e.g., one node in the train set and the other in the test set. We see that in *ogbn-arxiv*, a model trained in a transductive setting can exploit about $200,000$ edges, of which only 40% are in the training set. The phenomenon is less important in *ogbn-mag* where 80% of the edges are in the training set. Considering only the edges between labeled nodes, which account for a small portion of all the edges, *ogbn-papers100M* has also 80% of its edges coming from the training set.

Citation networks change over time, as fields grow in importance while others dwindle. This is shown in Table II which contains the most frequently occurring class per year in the *ogbn-arxiv* dataset. Some classes, such as the **cv** class (Computer Vision) were not present in the top 5 before suddenly reaching second then first position, as it happened between 2013 and 2015. These distribution changes emphasize the importance of inductive learning: GNNs need to be trained to be able to adapt to these changes.

[2]In the case of *ogbn-mag*, only the reverse edges related to a paper citing another paper are added.

V. Experiments

We compare the performance of models trained in two different settings: transductive and inductive learning. In the first setting, the model has access to the training data and the unlabeled validation and test data. In the second setting, the model has only access to the training data. Each setting follows the same temporal split presented in Section IV. A practical way to remove the edges going from or to unlabeled nodes is to generate the subgraph of the network that contains only the nodes within the given set.

The architectures we compare are the best performing architectures on the *ogbn-arxiv* dataset. The leaderboard can be found at https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-arxiv. Specifically, we train Deep RevGAT [6] and Adaptive Graph Diffusion Network (AGDN) [16] models. The Deep RevGAT architecture follows the message passing neural framework [9]. It is a GAT [7] which has been converted to a reversible GNN [6] to remove the memory constraints imposed by the storing of a feature matrix for each layer of a traditional GNN. We explore the importance of self-knowledge distillation [4] by training the Deep RevGAT with and without knowledge distillation. Additionally, we investigate the importance of feature selection, using either the original features, the features extracted with the GIANT framework, and the features with added label information.

All the experiments are performed with a 24 GB Nvidia RTX GPU. The code is written in Python, PyTorch and DGL (Deep Graph Library) [26]. The *ogbn-arxiv* dataset is taken from OGB [23]. Several Deep RevGAT are trained with a different number of layers. The AGDN model is trained both with and without using a big of tricks presented in [27] that

TABLE I
DISTRIBUTION OF EDGES IN TRAIN, VALIDATION AND TEST.

Graph	Detail	number of edges	edges in two groups
ogbn-arxiv	original graph	1,166,243	-
ogbn-arxiv	with self-loops and reverse edges	2,484,941	-
ogbn-arxiv	train nodes subgraph	829,007	-
ogbn-arxiv	validation nodes subgraph	86,671	-
ogbn-arxiv	test nodes subgraph	167,883	-
ogbn-arxiv	train and validation data	1,351,570	435,892
ogbn-arxiv	train and test data	1,710,834	713,944
ogbn-arxiv	validation and test data	506,098	251,544
ogbn-mag	original graph (only citations)	5,416,271	-
ogbn-mag	with self-loops and reverse edges	11,568,931	-
ogbn-mag	train nodes subgraph	8,389,507	-
ogbn-mag	validation nodes subgraph	177,111	-
ogbn-mag	test nodes subgraph	123,583	-
ogbn-mag	train and validation data	10,149,426	1,582,808
ogbn-mag	train and test data	9,575,822	1,060,732
ogbn-mag	validation and test data	533,884	231,190
ogbn-papers100M	original graph (only citations)	1,615,685,872	-
ogbn-papers100M	with self-loops and reverse edges	3,342,431,700	-
ogbn-papers100M	train nodes subgraph	21,315,319	-
ogbn-papers100M	validation nodes subgraph	330,045	-
ogbn-papers100M	test nodes subgraph	639,152	-
ogbn-papers100M	train and validation data	24,506,366	2,861,002
ogbn-papers100M	train and test data	25,144,635	3,190,164
ogbn-papers100M	validation and test data	1,583,153	613,956

TABLE II
TOP 5 CLASSES (BY SIZE) AND PER YEAR IN *obn-arxiv*. ONLY THE 5 MOST PROMINENT CLASSES ARE SHOWN.

Year	Class 1	size	Class 2	Size	Class 3	Size	Class 4	Size	Class 5	Size
2019	lg	8690 (21.88%)	cv	8584 (21.62%)	cl	4075 (10.26%)	it	2256 (5.68%)	ro	1602 (4.03%)
2018	cv	6846 (22.97%)	lg	4458 (14.96%)	cl	2849 (9.56%)	it	2273 (7.63%)	ai	1232 (4.13%)
2017	cv	4326 (20.18%)	it	2597 (12.11%)	lg	2114 (9.86%)	cl	1753 (8.18%)	ai	933 (4.35%)
2016	cv	2646 (16.19%)	it	2525 (15.45%)	lg	1374 (8.41%)	cl	1185 (7.25%)	ds	850 (5.20%)
2015	it	2210 (18.36%)	cv	1453 (12.07%)	lg	1008 (8.38%)	ds	736 (6.12%)	si	532 (4.42%)
2014	it	1755 (19.17%)	cv	705 (7.70%)	ds	631 (6.89%)	lg	593 (6.48%)	ni	483 (5.28%)
2013	it	1617 (19.88%)	ai	927 (11.40%)	lg	579 (7.12%)	ds	552 (6.79%)	ni	441 (5.42%)
2012	it	1316 (20.45%)	lg	680 (10.57%)	ai	547 (8.50%)	ds	433 (6.73%)	ni	341 (5.30%)
2011	it	1142 (25.80%)	ds	384 (8.67%)	lo	246 (5.56%)	ni	223 (5.04%)	ai	217 (4.90%)
2010	it	940 (26.37%)	ds	298 (8.36%)	lo	240 (6.73%)	ni	210 (5.89%)	ai	189 (5.30%)

can improve performance.

Each model is trained for 2,000 epochs. At the end, the model weights saved are the ones which led to the best validation accuracy. Additionally, each model is trained for 10 runs in order to mitigate the fluctuation in accuracy over each run.

Table III shows the results of the training computations in a transductive setting. The difference between validation and test accuracy is also shown in the last column. Table IV shows the results of the same models in the inductive setting.

We see that in the transductive learning setting, the Deep RevGAT model achieves the best accuracy with the smallest number of layers. Using self-knowledge distillation with more layers actually worsens the performance. The increase in the difference between validation and test accuracy indicates that the model is trying to fit the validation data, and that the validation data and test data are not completely similar. This is expected because the addition of new papers to the citation network may change the structure of the graph, and the model might not be able to predict these unforeseen structures.

In the inductive setting, the performance of the model with self-knowledge distillation increases with the number of layers, while the gap between validation and test accuracy decreases. The model with 5 layers and knowledge distillation achieves the best test accuracy of all the models trained with inductive learning. The performance is close to the one from the top model in the transductive setting.

Models trained on the same dataset in transductive and in inductive learning usually achieve better test accuracy in transductive learning [14]. This is an expected behaviour because the transductive model already had access to test

TABLE III
VALIDATION AND TEST ACCURACY FOR TRANSDUCTIVE LEARNING.

Model	Validation accuracy	Test accuracy	Difference between validation and test
RevGAT, teacher, 2 layers	76.99 ± 0.06	75.98 ± 0.12	1.01
RevGAT, KD 2 layers	77.13 ± 0.10	76.17 ± 0.15	0.96
RevGAT, 3 layers	77.13 ± 0.06	75.95 ± 0.11	1.18
RevGAT, KD 3 layers	76.93 ± 0.08	75.61 ± 0.15	1.32
RevGAT, 5 layers	77.11 ± 0.06	75.80 ± 0.12	1.31
RevGAT, KD, 5 layers	77.24 ± 0.09	75.97 ± 0.09	1.27
AGDN, original features *	-	73.46 ± 0.17	-
AGDN *with BoT, original features	-	74.10 ± 0.15	-

* the accuracy of these models were taken from the authors of the model.

TABLE IV
VALIDATION AND TEST ACCURACY FOR INDUCTIVE LEARNING. THE BEST SCORE IS HIGHLIGHTED IN BOLD. KD STANDS FOR KNOWLEDGE DISTILLATION.

Model	Validation accuracy	Test accuracy	Difference between validation and test
RevGAT, 2 layers,	76.75 ± 0.08	75.91 ± 0.14	0.84
RevGAT, KD, 2 layers,	76.74 ± 0.05	75.81 ± 0.13	0.93
RevGAT, 3 layers,	76.78 ± 0.08	76.04 ± 0.10	0.74
RevGAT, KD, 3 layers,	76.78 ± 0.09	75.97 ± 0.09	0.81
RevGAT, 5 layers,	76.84 ± 0.05	76.00 ± 0.10	0.84
RevGAT, KD, 5 layers,	76.85 ± 0.06	$\mathbf{76.07 \pm 0.07}$	0.78
RevGAT, 3 layers, no label features,	76.54 ± 0.09	75.83 ± 0.10	0.71
RevGAT, KD, 3 layers, no label features,	76.63 ± 0.08	75.97 ± 0.06	0.66
AGDN, original features,	72.47 ± 0.12	73.20 ± 0.17	-0.73
AGDN with BoT, original features,	73.62 ± 0.07	74.00 ± 0.07	-0.38
AGDN, GIANT features,	76.28 ± 0.17	75.38 ± 0.22	0.9

data in the training phase. The fact that the models achieve similar performances in transductive and inductive settings may be an indication that the models are not fully exploiting the information available during training, and that new architectures are needed to leverage it. Indeed, the fact that the AGDN model trained with the original features has a higher test accuracy than its validation accuracy suggests that the model is slightly underfitting the data, and that either the features must be changed, or the model needs to be modified to better exploit the data.

The use of features that exploit the graph topology, created with the GIANT framework, could partly explain what the gap between transductive and inductive learning is small: because the features are created using the entire graph, some of the test information is already embedded in the features of the training nodes. Since the AGDN model trained with the original features achieves comparable accuracy in both settings, the GIANT extracted features cannot account for the small gap.

In the inductive setting, directly encoding the label information as part of the nodes features using a one-hot encoding scheme does not seem to improve performance. While the Deep RevGAT model with 3 layers trained with label features performs better than the model trained solely on the GIANT features, adding self-knowledge distillation eliminates the difference.

The most likely reason why the models achieve comparable results in transductive and inductive settings is that each model employs some form of neighbourhood sampling in the training phase. This makes the models more robust and acts as a regularizer in preventing overfitting to the data. The importance of edges coming from validation and test data is reduced as they do not systematically appear in the sampled edges.

VI. CONCLUSION

In this paper we studied the difference between transductive and inductive learning for citation networks where the graphs have been converted to undirected graphs and the train, validation and test sets have been constructed using the temporal information of the nodes. We saw that the training datasets contain many edges from the validation and test sets: one fifth of the edges in the train set of *ogbn-mag* are from the validation or test set, while two thirds of the edges in *ogbn-arxiv* come from outside of the training set.

To analyze the importance of these edges, we trained state of the art GNNs in an inductive setting where the training set contains only edges from within the train set. We found

that the networks achieved similar performances in either a transductive or inductive setting. This may be due to the neighbourhood sampling methods used by most GNNs that reduce overfitting to the data. This also suggests that current models can be improved, as models usually perform better in transductive settings.

Furthermore, we observed that the gap between validation and test accuracy was lower in the inductive setting than in the transductive setting. This is an indication that the models trained in the inductive setting were better at generalizing than their counterparts trained in the transductive setting. Moreover, this suggests that new architectures will likely be able to improve on the current results.

Some perspectives for future work include performing the same experiments on *ogbn-papers100M* and *ogbn-mag* to explore whether the models maintain their accuracy when we remove the edges from different data splits.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[4] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 3712–3721.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[6] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 6437–6449.

[7] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[8] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, ser. JMLR Workshop and Conference Proceedings, M.-F. Balcan and K. Q. Weinberger, Eds., vol. 48. JMLR.org, 2016, pp. 40–48.

[9] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 1263–1272.

[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[11] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 1025–1035.

[12] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2019, pp. 257–266.

[13] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal Graph Networks for Deep Learning on Dynamic Graphs," *arXiv:2006.10637 [cs, stat]*, Oct. 2020.

[14] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[15] E. Chien, W.-C. Chang, C.-J. Hsieh, H.-F. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon, "Node Feature Extraction by Self-Supervised Multi-scale Neighborhood Prediction," *arXiv:2111.00064 [cs]*, Oct. 2021.

[16] C. Sun and G. Wu, "Adaptive Graph Diffusion Networks with Hop-Wise Attention," *arXiv:2012.15024 [cs]*, Dec. 2020.

[17] C. Sun, H. Gu, and J. Hu, "Scalable and Adaptive Graph Neural Networks with Self-Label-Enhanced training," *arXiv:2104.09376 [cs]*, Jul. 2021.

[18] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015.

[19] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky, "Graph normalizing flows," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 13 556–13 566.

[20] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs Go As Deep As CNNs?" in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 9266–9275.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013.

[22] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective Classification in Network Data," *AI Magazine*, vol. 29, no. 3, p. 93, Sep. 2008.

[23] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin, Eds., 2020.

[24] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs," Oct. 2021.

[25] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia, "Microsoft Academic Graph: When experts are not enough," *Quantitative Science Studies*, vol. 1, no. 1, pp. 396–413, Feb. 2020.

[26] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[27] Y. Wang, J. Jin, W. Zhang, Y. Yu, Z. Zhang, and D. Wipf, "Bag of Tricks for Node Classification with Graph Neural Networks," *arXiv:2103.13355 [cs]*, Jul. 2021.