A Study on Active Learning for Graphs

Pratheeksha Nair
pratheeksha.nair@mail.mcgill.ca
McGill Univesity
Montreal, Quebec, Canada

Zhi Wen
zhi.wen@mail.mcgill.ca
McGill Univesity
Montreal, Quebec, Canada

ABSTRACT

This is a comprehensive study of active learning methods on real-world graph datasets. Active Learning for interconnected data has been studied over the years and has gained increasing importance in recent times, especially due to its applications in tasks where labeling is laborious and requires human experts, such as drug discovery and protein-protein interaction prediction. In this work, various active learning strategies are compared across 6 real-world datasets from different domains for the task of node classification. The main goal of this evaluation is to benchmark a range of active learning strategies against state-of-the-art and identify the ones that consistently perform well. We also propose a simple strategy for selecting nodes for training the node classifier, and our experiments show promising results of this strategy.

CCS CONCEPTS

• **Active Learning**; • **Network data**; • **Human-in-the-loop learning**; • **Graphs**;

KEYWORDS

Real-world datasets, Graph Neural Networks, Node Classification, Active Learning

1 INTRODUCTION

Active Learning is a sub-field of Machine Learning that is based on the hypothesis that a model will perform better with less training if it is allowed to choose the data to learn from [17]. Supervised learning is a popular technique used to train Machine Learning models that are often deployed in multiple real-world applications. In supervised classification problems, data instances with ground truth labels are used for training a model that can predict the labels of unseen data instances. Therefore, the performance of a supervised learning models depends on the quality and quantity of training data. While there exist a multitude of Machine Learning tasks where labelled training instances can be easily obtained in hundreds and thousands, for example the detection of spam e-mails, there are several more sophisticated supervised learning tasks where labelling is both expensive and done manually [17]. Active Learning tries to overcome the bottleneck of labelling by choosing the most relevant and informative samples for training. This is done by *querying* an *oracle*, such as a human or an annotator, for the labels of the selected samples. The goal of the active learner is to maximise accuracy using a *budget* of labeled instances, thus maintaining a fixed annotation cost. Such a strategy can also prevent the leaner from being overwhelmed with uninformative or redundant samples.

In recent times, graphs are being ubiquitously used for encoding relational data and graph datasets have become extremely popular. Learning effective representations of graphs has become critical in many applications [6]. Graph Neural Networks have been extensively employed in node classification [7, 18] and link prediction [2, 21]. However they require a large amount of labelled data for training [6]. This is where Active Learning, a promising strategy to address this problem, comes into the picture.

2 MOTIVATION

Active Learning strategies are most useful when unlabelled data is abundant and manual labelling is expensive. Some examples where active learning methods are widely used include[17],

(1) The accurate labeling of speech utterances for speech classification as this requires trained linguists and is time consuming
(2) Detailed annotation of documents for information extraction tasks which may require experts in various subjects
(3) Labeling of individual documents or media files in specific fields for classification and filtering tasks

More specific to graphs and network data, the training instances (nodes in the graph) are connected by links and their labels are often correlated [9]. Thus, these links can be useful while selecting the most informative samples. For example, nodes connected to each other have a higher likelihood of having the same label than unconnected nodes.

Some examples of cases where active learning on graphs is useful include,

(1) predicting the effects of new substances on organisms in biological networks[4]
(2) predicting effects of proteins on other biomolecules in molecular networks[12]
(3) identifying a web of individuals involved in a criminal activity based on the evidence of their connections[15]

To summarize, the problem of active learning for interconnected data is well motivated by multiple use cases, particularly when link information is readily available and labeling is expensive and/or requires human attention.

The main motivation of this study is to act as an easy-to-understand guide for early researchers interested in Active Learning for graphs. The advantages of this study may be narrowed down to the following.

COMP 596 '20, Nov 30, 2020, McGill, Montreal

Dataset	Nodes	Classes	Features	Avg. Deg.	Avg. CC	Homophily
Citeseer	2110	6	3703	2.84	0.17	-0.077
PubMed	19,717	3	500	6.34	0.06	-0.043
Amazon Computers	13,752	10	767	36.74	0.35	-0.056
Disease	1044	2	1000	2.0	0.0	-0.544
Wiki-CS	11,701	10	300	36.94	0.47	-0.092
PPI	8281	121	50	31.8	0.18	-0.046

Table 1: Summary of datasets. Disease dataset was obtained from https://github.com/HazyResearch/hgcn. The Github dataset was obtained from https://github.com/benedekrozemberczki/MUSAE. All the other datasets were taken from Pytorch Geometric library.

(1) It provides insights on what kind of Active Learning strategies are best suited for a given type of network dataset.
(2) It establishes baselines for quick comparisons with new methods.
(3) It estimates the optimal budget size for querying an oracle, depending on the network type and Active Learning strategy.

3 RELATED WORK

All active learning strategies involve evaluating the informativeness of unlabeled samples (either generated newly or sampled from a distribution) [17]. One of the most commonly used query framework is *uncertainty sampling*[9] where the active learner queries the data samples which is it most uncertain of how to label. Most general uncertainty sampling strategies use *entropy* of label predictions as a measure of uncertainty [17].

Many active learning algorithms employ a greedy strategy to choose data samples that maximise a combination of the entropy measure and some kind of graph property like the degree centrality score, information density score or clustering co-efficient [1, 9].

Another class of active learning strategies that have come into focus more recently include methods that pose this problem as one of exploration versus exploitation, using a Reinforcement Learning approach. Fang et al.[5] use deep Q-networks to learn active learning policies for the problem of named entity recognition. Liu et al.[8] propose a solution for neural machine translation using active learning based on reinforcement learning. While these methods were proposed for i.i.d data samples, Graph Policy Network [6] was proposed for transferable active learning on graphs. This work poses the problem as a Markov Decision Process and learns an optimal query strategy using REINFORCE[19].

A more recent work, quite similar to this proposed project is an evaluation of active learning methods for node classification[9]. Here, the performance of various active learning strategies (discussed above) are compared across multiple real-world datasets for the task of node classification. While this serves as a good reference, the present study differs from it by including active learning methods more specific to interconnected data for comparisons. They also do not consider the more recent Reinforcement Learning based approaches [6], which is studied in this work.

4 PROBLEM DEFINITION

The main goals of this project are *to study the performance of various existing active learning methods for different real-world graph datasets for node classification tasks.*

More specifically, this study attempts to answer the following questions:

(1) How do various active learning strategies perform on different graph datasets?
(2) Is there a strategy that consistently performs well/poorly, and what is the possible reason?

Conducting this evaluation will help provide insights on what kind of an approach makes most sense for graphs of a particular nature. For example, Madhawa et. al [9] empirically show that for graphs with higher level of clustering, sampling nodes with highest clustering coefficients is a good strategy. This can also provide strong baselines to compare against while developing new methods.

5 DATASETS

In order to carry out an extensive study, the performance of all the algorithms is compared on 6 real-world datasets from different domains such as citation networks, product networks, co-author networks, biological networks and social networks. This setup was adopted from [9]. A summary of the 6 chosen datasets is in Table 1.

CiteSeer and PubMed[16] are commonly used citation graphs. The nodes are documents and edge between two nodes indicate that they have cited each other. The bag-of-words features of the text content of a document are the node features.

Amazon Computers is a subgraph of the Amazon co-purchase graph[10]. Products are represented as nodes, and two nodes are connected if they are frequently bought together. Node attributes are bag-of-words features of product reviews. The product category is used as the node label.

The disease dataset[3] is a disease propagation network. The label of a node indicates whether it is infected or not and the features indicate the susceptibility to the disease.

The Wiki-CS dataset[11] has nodes as Wikipedia articles about computer science. An edge exists between two nodes if one article has a hyperlink to the other. GloVe word embeddings[14] of the text content of an article is used as the feature vector of its node.

The protein–protein interaction (PPI) graph represents interactions between proteins in human brains, blood, and kidneys[13, 22]. Protein properties are used as node attributes in a PPI graph.

6 METHODOLOGY

This work uses various active learning strategies for sampling training nodes to train an SGC node classifier. These strategies include random sampling, greedy strategies (optimizing a given objective), graph embedding methods and reinforcement learning based models.

The training framework that involves active learning strategies in our study includes the following parts:

(1) Use a sampling strategy for selecting informative nodes for training a classifier model. The total number of nodes picked is equal to the budget size provided. Node selection can either be done in one pass (for strategies that only rely on the graph), or iteratively as the classifier model is being trained (for strategies that also depend on models' output).
(2) Train a node classifier model using the above train set and evaluate its performance on a held-out test set.
(3) Run the above experiment for a particular dataset using different data splits and report the average AUPRC (area under the precision recall curve) for different budget sizes. The AUPRC metric is popularly used for evaluating classifier performances on imbalanced test sets.

Each of the above 3 parts is varied to study their impact. Changing the sampling strategy helps identify the best Active Learning method for a specific dataset. Varying the classifier model, e.g, GCN, SGC, GAT compares the classification performance for different node classifiers. Experimenting on different datasets provides insights on the kind of graph properties to consider while choosing Active Learning strategies. For the scope of this work, the sampling strategies and datasets are varied while using a single, powerful node classifier model.

Given a fixed budget (k), we compare the following sampling methods. Methods 1-4 are treated as baselines, method 5 is a new strategy that we propose and methods 6-7 are state-of-the-art Active Learning algorithms.

(1) **Random**: We sample k nodes uniformly at random from the train set.
(2) **Entropy**: After each time the classifier weight is updated, calculate the entropy (uncertainty) of the predictions made by the current model over the unlabelled nodes (from the train set) and choose the node which maximizes the entropy iteratively till budget is reached
(3) **Clustering coefficient**: We sample the top k nodes that have maximum clustering coefficient from all the training nodes in the graph.
(4) **Degree**: Pick the top k nodes that have maximum degree centrality measures from among all the training nodes in the graph. This serves as a baseline to the following strategy.
(5) **Clustering+Degree**: In this strategy, we cluster the graph and sample the node with highest degree in the subgraph obtained by the nodes of each cluster. More specifically, we sort the clusters by their sizes from the largest to the smallest, and within each cluster we sort the nodes by their degrees from the highest to the lowest. Then we iteratively sample one node in each cluster, in the order given above, until the budget is met. The graph clusters provide groups of closely related nodes and picking the node with the highest degree within a cluster, gives the most informative node within a group of similar nodes. Sampling from each cluster thus provides a good sample of informative nodes from the entire graph.
(6) **AGE** [1]: After each time the classifier weight is updated, compute each node's PageRank score, its distance to the closest k-means cluster center, and its prediction entropy, and select the node which maximizes a linear combination of the quantiles of these three quantities. The linear weights can be either uniform, i.e. $1/3$ for each quantile, which we denote as *static* AGE, or varies as the training proceeds, which we denote as *adaptive* AGE. Detailed discussions on the linear weights are in Appendix B.
(7) **GPA** [6]: Graph Policy network for transferable Active Learning, learns a transferable policy of selecting nodes for training a GNN node classifier. It treats the problem as a sequential decision process on graphs and trains another GNN model, which acts as the policy network, with reinforcement learning to learn the optimal query strategy. At each time-step, the graph state is a matrix of node state representations which is a concatenation of node properties. These properties include degree, entropy of the label distribution predicted by a classification GNN, the divergence between a node's predicted label distribution and its neighbouring nodes'. The action is to select a node for querying and the reward is the classification GNN's performance on the validation set. We refer the reader to [6] for more details on the implementation.

7 EXPERIMENT SETUP

We run node classification experiments[1] on the 6 datasets mentioned in Table 1. We use the SGC graph neural network model [20] for our experiments on node classification. SGC is a simplified GNN architecture that does not include a hidden layer and nonlinear activations which has been shown to give better results over GCN [20]. We tune the hyperparameters for each dataset with the random sampling strategy and a budget of 50.

For each of the datasets, we split them into train, validation and test sets. We then experiment with various Active Learning strategies of sampling nodes from the train set and use the sampled nodes for training the SGC model. We report the AUPRC of the model on the test set as an average over 5 runs using different seeds for both splitting the data and initializing the SGC model parameters. For these experiments the train-test-validation split is $60\% - 20\% - 20\%$.

In the GPA paper, the authors use GCN instead of SGC, and they evaluate in terms of F1 scores instead of AUPRC. Furthermore, GPA evaluates only on a subset of our datasets, and they use different data splits than ours. Therefore, to ensure a fair comparison, we conducted further experiments to mimic GPA's setup, by using the same validation and test sizes, using GCN in addition to SGC, and evaluating with F1 scores.

8 RESULTS AND DISCUSSION

The comparison of the performance of the all the Active Learning strategies mentioned Section 6, except GPA, is shown in Figure 1. We see that on all datasets except Amazon, the proposed strategy

[1] Our code is available at https://github.com/nair-p/AL4G

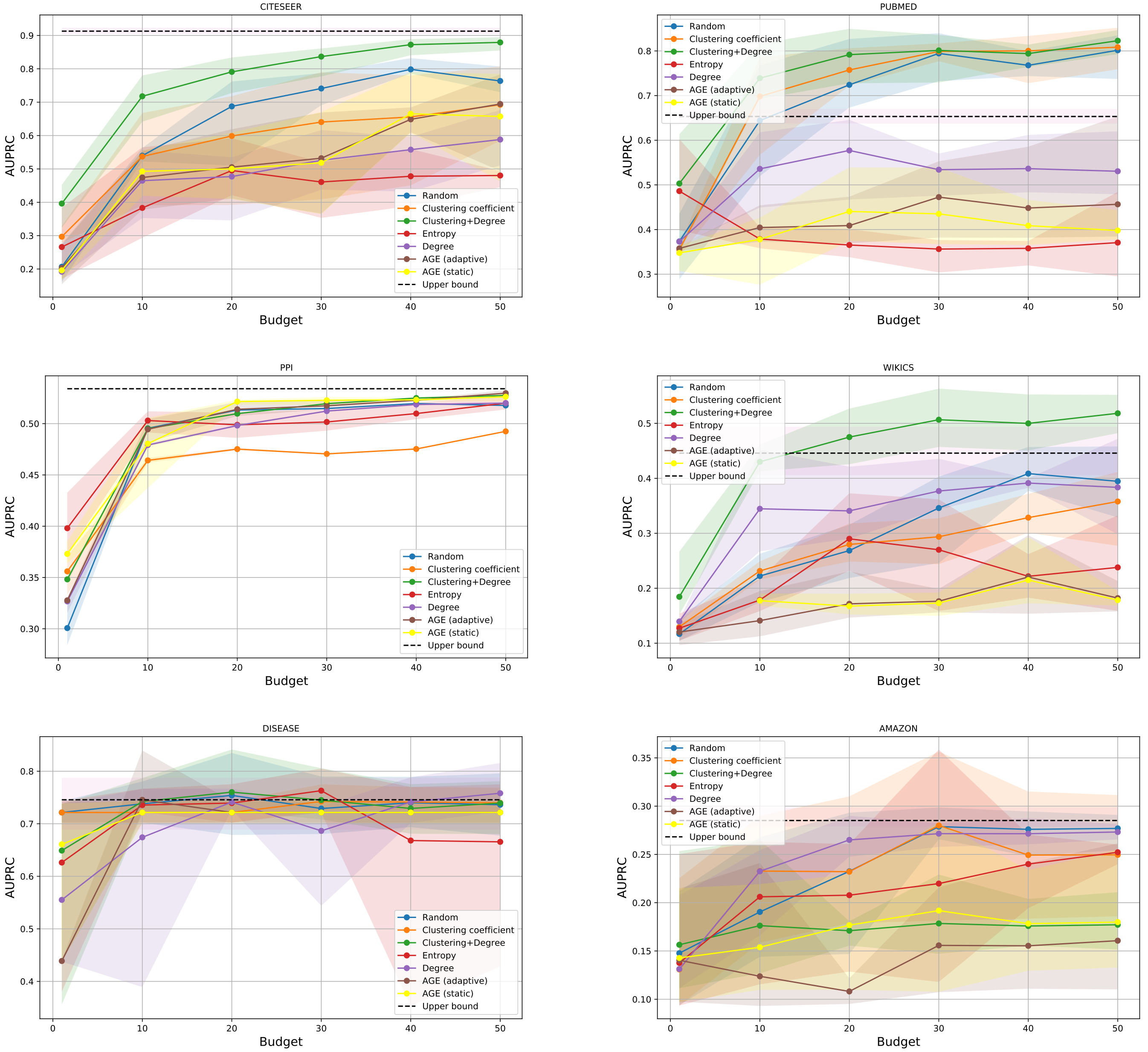

Figure 1: AUPRC on different datasets. The title of the plots indicate the dataset. "Upper bound" indicates the results obtained by training on the entire train set (vanilla semi-supervised setting).

(Clustering + Degree) either outperforms or performs as well as the other methods.

For both Citeseer and Pubmed, Clustering + Degree sampling works best for all budget sizes. Entropy sampling performs the worst. We hypothesize that the graph clusters loosely separate the nodes by their classes, and sampling from each of the clusters encourages the train set to contain at least one sample from each class. Moreover, choosing the node with the highest degree from within the cluster works well probably because majority of the nodes connected to this node are likely to have the same label (due to homophily).

For Amazon, noting that it has a very high average degree assortativity (see Table 1), i.e a high degree node is very likely to connect with other high degree nodes, picking the highest degree nodes from clusters may result in losing out on important, connected nodes. We suspect this might be the reason that the proposed method (Clustering+Degree) performs poorly on this dataset.

Table 2: Mean F1 scores of active learning algorithms on Pubmed and Citeseer. Highest score in each column is in bold.

Algorithms	Pubmed		Citeseer	
	Micro-f1	Macro-f1	Micro-f1	Macro-f1
Random	0.53	0.41	0.66	0.65
Clustering Coeff.	0.51	0.43	0.61	0.55
Clustering+Deg	0.49	0.32	**0.72**	**0.71**
Entropy	0.42	0.24	0.45	0.41
Degree	0.57	0.42	0.51	0.42
AGE (static)	0.42	0.22	0.52	0.49
AGE (adaptive)	0.41	0.26	0.42	0.35
GPA (in paper)	**0.78**	**0.76**	0.66	0.57

Furthermore, comparison with GPA is shown in Table 2. Each score is the highest of the scores received by GCN and SGC in the given setting, and the full results that include both GCN and SGC are shown in Table 3. First, we observe that, despite our efforts to replicate GPA's evaluation scheme, our random baseline performs worse on Pubmed compared to what is reported in the GPA paper, whereas on Citeseer our random baseline performs better. We suspect this is due to different GCN implementations or data processing steps.

Second, we observe that on Citeseer, Clustering + Degree outperforms GPA's reported performance substantially, in terms of both micro-f1 score and macro-f1 score. Even considering the confounding fluctuations as indicated by the discrepancy in the random baseline, it is still very likely that Clustering + Degree is similar to, if not better than, GPA on Citeseer, if under the exact same evaluation scheme. This is remarkable especially considering that Clustering + Degree is a static, parameter-free sampling strategy, as opposed to GPA which employs an RL agent that is trained on two other datasets. While our experiments do not provide conclusive results, we believe they definitely warrant further study of using Clustering + Degree.

9 CONCLUSION AND FUTURE WORK

This paper is a study on active learning methods for graphs and it compared the performance of 7 different Active Learning strategies on 6 real-world network datasets. One of our contributions is we provide a benchmark for common active learning strategies on the task of node classification, which serves to provide comparison for future active learning strategies. We also propose a new active learning strategy that favors the highly connected nodes in graph clusters. Our experiments show promising results of this new method, which outperforms or is at par with other popularly used strategies, including state-of-the-art methods, such as entropy, AGE and GPA.

REFERENCES

[1] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2017. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085* (2017).

[2] Zhu Cao, Linlin Wang, and Gerard De Melo. 2018. Link prediction via subgraph embedding-based convex matrix completion. In *Thirty-Second AAAI Conference on Artificial Intelligence.*

[3] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Advances in neural information processing systems.* 4868–4879.

[4] Hyunghoon Cho, Bonnie Berger, and Jian Peng. 2016. Reconstructing causal biological networks through active learning. *PloS one* 11, 3 (2016), e0150611.

[5] Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383* (2017).

[6] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. 2020. Graph Policy Network for Transferable Active Learning on Graphs. *arXiv preprint arXiv:2006.13463* (2020).

[7] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[8] Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning.* 334–344.

[9] Kaushalya Madhawa and Tsuyoshi Murata. 2020. Active Learning for Node Classification: An Evaluation. *Entropy* 22, 10 (2020), 1164.

[10] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval.* 43–52.

[11] Péter Mernyei and Cătălina Cangea. 2020. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks. *arXiv preprint arXiv:2007.02901* (2020).

[12] Thahir P Mohamed, Jaime G Carbonell, and Madhavi K Ganapathiraju. 2010. Active learning for human protein-protein interaction prediction. *BMC bioinformatics* 11, S1 (2010), S57.

[13] Rose Oughtred, Chris Stark, Bobby-Joe Breitkreutz, Jennifer Rust, Lorrie Boucher, Christie Chang, Nadine Kolas, Lara O'Donnell, Genie Leung, Rochelle McAdam, et al. 2019. The BioGRID interaction database: 2019 update. *Nucleic acids research* 47, D1 (2019), D529–D541.

[14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).* 1532–1543.

[15] Reihaneh Rabbany, David Bayani, and Artur Dubrawski. 2018. Active search of connections for case building and combating human trafficking. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2120–2129.

[16] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[17] Burr Settles. 2009. *Active learning literature survey.* Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[19] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[20] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. 2019. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153* (2019).

[21] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems.* 5165–5175.

[22] Marinka Zitnik and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33, 14 (2017), i190–i198.

Table 3: Mean F1 scores of active learning algorithms on Pubmed and Citeseer. Highest score in each column is in bold. GPA results are from their paper[6]. The × indicates that the method was not implemented using that model, i.e GPA was not tested using an SGC classifier model in [6]

Algorithms	**Pubmed**				Citeseer			
	Micro-f1		Macro-f1		Micro-f1		Macro-f1	
	SGC	GCN	SGC	GCN	SGC	GCN	SGC	GCN
Random	0.41	0.53	0.23	0.41	0.66	0.50	0.65	0.43
Clustering Coeff.	0.43	0.51	0.31	0.44	0.61	0.52	0.56	0.43
Clustering+Deg	0.49	0.45	0.32	0.30	**0.72**	0.44	**0.71**	0.37
Entropy	0.42	0.43	0.24	0.28	0.46	0.44	0.41	0.35
Degree	0.57	0.52	0.42	0.36	0.46	0.52	0.42	0.42
AGE (static)	0.38	0.42	0.21	0.22	0.53	0.38	0.50	0.29
AGE (adaptive)	0.40	0.42	0.19	0.26	0.43	0.39	0.35	0.29
GPA (in paper)	×	**0.78**	×	**0.76**	×	0.66	×	0.57

A ADDITIONAL RESULTS

Additional results on comparison with GPA, including both SGC and GCN models, are shown in Table 3.

B AGE LINEAR WEIGHTS

We found that the linear weights calculation in the original AGE paper [1] is different from the actual implementation. Specifically, in the paper, the weight for the PageRank quantile, γ_t, is drawn from a Beta distribution $\gamma_t \sim Beta(1, n_t)$, while the other two weights are drawn from $Beta(1, n_t')$, where n_t and n_t' are two undefined variables that increase and decrease as the number of iterations increases, respectively. And finally, the weights are normalized to sum to 1.

However, after examining the code released by the authors[2], we found that γ_t is drawn from $Beta(1, 1.005 - c^t)$, where c is a dataset-specific hyperparameter and t is the number of iterations, and α_t and β_T are determined by $(1 - \gamma_t)/2$.

Therefore, because of this inconsistency in the paper and in the actual implementation, and of our decision to use SGC in addition to GCN, we decided to follow the spirit of the original paper and design our own weight computation adaptation.

The spirit is that the quantities that depend on the model, i.e. entropy and k-means distances, should have small weights at first, and have larger weights as the model is being trained. As a result, if the classifier is GCN, as is in the AGE paper, both α and β should increase, and thus we calculate them as $\alpha_t = \beta_t = t/150$, since the largest budget in our experiments is 50. Similarly, if the classifier is SGC, only α is increased, since the node embeddings are not updated in SGC, and α is calculated as $\alpha_t = t/150$. Finally, $\beta_t = \gamma_t = (1 - \alpha_t)/2$.

[2] https://github.com/vwz/AGE