A Critical Look at the Evaluation of GNNs under Heterophily: Are We Really Making Progress?

Anonymous ACL submission

Abstract

It has been a strongly held belief that GNNs only perform well on graphs with homophily. However, the paper showed that it's not the case. It's the datasets themselves that are problematic. We are reproducing a subset of the results from the paper by considering 3 models: GAT-sep, GT-sep and FSGNN. We are then evaluating their performance on these datasets.

1 Introduction

The question the paper (Platonov et al., 2023) sought to answer was: ”Do GNNs perform well under heterophily“? To answer this, the paper did the following:

1. Evaluation of current heterophily datasets.
2. Evaluation of both heterophily specific and non-heterophily specific GNN architectures under the old benchmark datasets.
3. Evaluation of both heterophily specific and non-heterophily specific GNN architectures under the new benchmark datasets.

1.1 Metrics for Homophily

Three most commonly used metrics for measuring homophily are given in the sections below.

1.1.1 Node Homophily

Node homophily is simply defined as the proportion of neighbours that have the same class for each node. The proportion is calculated for each node, and is then averaged across all the nodes.

$$H_v = \frac{|u \in N(v) : C_u = C_v|}{|N(v)|}$$

where C_v is the class or label of node v, $N(v)$ is the set of neighbors of node v and $|N(v)|$ is the number of neighbors of node v.

The overall homophily of the graph would then be:

$$H_{graph} = \frac{1}{|V|} \sum_{v \in V} H_v$$

1.1.2 Edge Homophily

Edge Homophily is defined as the fraction of edges that connect nodes of the same class.

If we take the total number of edges in the network as E, the number of edges that connect nodes of the same type to be E_{same}, then edge homophily would be given as:

$$h_{edge} = \frac{E_{same}}{E}$$

1.1.3 Adjusted Homophily

A more sophisticated understanding of homophily in networks can be obtained by utilising the idea of adjusted homophily, particularly in situations where the network's underlying node type distribution may have an impact on the basic homophily metric. It accounts for the default probability of connections forming between comparable nodes based only on the network's structure. It is calculated as:

$$h_{adj} = \frac{h_{edge} - \sum_{k=1}^{C} D_k^2/(2|E|)^2}{1 - - \sum_{k=1}^{C} D_k^2/(2|E|)^2}$$

Here, h_{edge} is the edge homophily, C is the number of classes in the graph, D_k is the degree sum for nodes of type k and $|E|$ is the total number of edges in the network.

2 Architectures

We took the following subset of models from the original paper to evaluate:

1. **GAT-sep**: The Graph Attention Network (GAT) model (Veličković et al., 2018) incorporates an addition "attention" parameter in

the model. It works on the assumption that all neighbours are not equally important as is the case in other architectures like GCN. The paper uses a variant of GAT, which they termed as "GAT-sep". This is the traditional GAT model with added self loops.

2. **GT-sep**: The Graph Tranformer model introduced the transformer architecture (previously used heavily in NLP (Vaswani et al., 2017)) to arbitrary graph structures(Shi et al., 2020).

3. **FSGNN**: The Features Selection Graph Neural Network (FSGNN) model(Maurya et al., 2021) takes a different approach. It feeds all computed features to the model with a weight parameter for each feature with the hope that the model itself will be able to learn which features are important.

For more description of these architectures, see appendix.

2.1 Datasets

In line with the paper's methodology, we made the following datasets pluggable. First, we have the existing datasets:

1. **Cornell,Texas and Wisconsin** [1]: Three sub-datasets of the WebKB1 webpage dataset. It contains the data collected from the computer science departments of these 3 universities. The nodes in the dataset are web-pages and edges are hyperlinks between these pages. The target field is the web page category and falls into one of the following: student,project,course,staff or faculty. The authors identified the following problems with the dataset: the dataset size is too small and the classes are imbalanced as well.

2. **Squirrel and Chameleon** [2]: The nodes represent page-page networks on chameleons and squirrels. The nodes represent articles and the edges represent hyperlinks between them. However, both of these datasets have a high number of duplicate nodes (e.g. the squirrel dataset, there is a group of 48 nodes that has the same regression target and the same 15 neighbours that appear in train,test and validation datasets).

The paper, after identifying these problems, stated that any dataset used to benchmark GNNs under heterophily should satisfy the following three requirements:

- The datasets should be heterophilous
- The dataset should have a structure that's suited for graph learning tasks.
- The datasets should be diverse i.e. they should come from various domains.

The authors then proposed the following datasets as new benchmark datasets [3]:

1. **Roman Empire**: This dataset is based on the Wikipedia page of Roman Empire. Each node corresponds to a non-unique word in the article. Two words are connected with an edge if either these words follow each other in the text or if those words are connected in the dependency tree of the sentence. The goal is to classify a node on its syntactic role (total 18 syntactic roles like subject, direct object etc.). The graph is chain like in its structure.

2. **Amazon Ratings**: Based on the Amazon product co-purchasing network metadata. The nodes in the dataset are products and two products are connected by an edge if they are frequently bought together. The task is to predict the rating of the product (value of 1-5).

3. **Minesweeper**: The graph in Minesweeper is a 100x100 grid where each node is a cell and two nodes are connected by an edge if they are adjacent in the grid. 20% of nodes are randomly chosen as mines. The task is to predict which nodes are mines.

4. **Tolokers**: The data is sourced from the Toloka crowdsourcing platform. The nodes represent users ("tolokers") that have worked in at least one of the 13 projects. Two users are connected by an edge if they have worked on the same project. The goal is to predict which users have been banned in one of the projects.

[1] https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.WebKB.html

[2] https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.WikipediaNetwork.html

[3] https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.HeterophilousGraphDataset.html

5. **Questions**: The dataset has contains data sourced from the Yandex Q platform. The nodes are users and an edge connects two users if one user answered another user's question during a 1-year time frame (Sep. 2021-Aug. 2021). The task here is to predict which users remained active at the end of the period. The features were taken as the words in each user's profile description. However, $\tilde{1}5\%$ of users did not have profile descriptions. So, an additional field was added to the embeddings to indicate the absence of a description.

3 Experiments

To test the accuracy, we ran the models on the datasets mentioned above.

3.1 Training and validation Hyper parameters

For both GT-sep and GAT-sep, we took momentum as 0.1 and the dropout rate as 0.2. The optimizer used was Adam(Kingma and Ba, 2017) with a learning rate of `3e-5`.

For FSGNN, we took LR as 0.001 and weight decay as 1e-5.

3.2 Overall Performance

On the older datasets, the model performance is shown in the table below:

-	squirrel	squirrel-filtered	chameleon	chameleon-filtered
GAT-sep	43%	42%	55%	35%
GT-sep	40%	33%	49%	34%
FSGNN	67%	35%	77%	36%

Table 1: Performance on older datasets

On the newly proposed datasets, the performance is shown in the table below:

	roman-empire	minesweeper	amazon	questions	tolokers
GAT-sep	85%	93%	51%	71%	84%
GT-sep	83%	92%	51%	71%	84%
FSGNN	70%	81%	44%	–%	79%

Table 2: Performance on proposed datasets

3.3 References

References

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2021. Improving graph neural networks with simple architecture design.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at the evaluation of gnns under heterophily: are we really making progress?

Yunsheng Shi, Zhengjie Huang, Wenjin Wang, Hui Zhong, Shikun Feng, and Yu Sun. 2020. Masked label prediction: Unified massage passing model for semi-supervised classification. *CoRR*, abs/2009.03509.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs.

Acknowledgements

We would like to express our deepest gratitude to Professor Vinti Agarwal, our instructor for the Graph Mining class at BITS Pilani. Her guidance and expertise have been invaluable throughout the course, providing us with a solid foundation in the field.

A Model Architectures

A.1 GAT-sep

The Graph Attention Network (GAT) model incorporates an addition "attention" parameter in the model. It works on the assumption that all neighbours are not equally important as is the case in other architectures like GCN. The paper uses a variant of GAT, which they termed as "GAT-sep". This is the traditional GAT model with added self loops.

Usually, in GNNs, the message computation is done as:

$$h_u^{(l)} = \sigma(\sum_{v \in N(u)} \frac{W^l h_v^{(l-1)}}{|N(u)|})$$

This means that each neighbouring node is given the same weight i.e. $\frac{1}{|N(v)|}$. Where GAT-sep differs is, that the weight for each neighbouring node is also a model parameter. This is given by:

$$h_u^{(l)} = \sigma(\sum_{v \in N(u)} \alpha_{uv} W^l h_v^{(l-1)})$$

How is α_{uv} computed?

The computation of α_{uv} depends on the computation of another quantity e_{uv}. e_{uv} is calculated as:

$$e_{uv} = \phi_{MLP}(FLATTEN(W^{(l)}h_u^{(l-1)}, W^{(l)}h_v^{(l-1)}))$$

Then, α_{uv} is computed as:

$$\alpha_{uv} = \frac{exp(e_{uv})}{\sum_{k \in N(u)} exp(e_{uk})}$$

The GAT-sep model differs from the traditional GAT model w.r.t. the additional term for separation of self embeddings. So, the message computation step becomes:

$$h_u^{(l)} = \sigma(\sum_{v \in N(u)} \alpha_{uv} W^l h_v^{(l-1)} + B^{(l)} h_u^{(l-1)})$$

The paper cites Zhu as the reason for introducing this additional parameters(Zhu et al., 2020).

A.2 GT-sep

The Graph Tranformer (GT) architecture is an extension of the transformer architecture to graph datasets. The architecture is shown in the figure below:

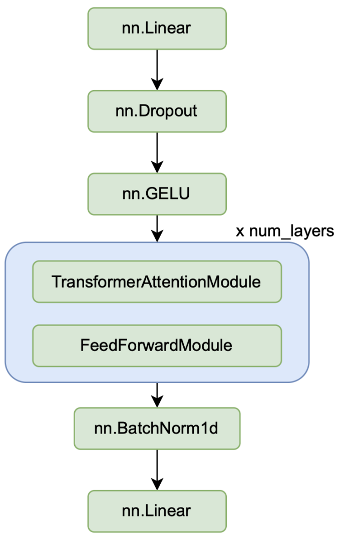

Figure 1: GT Architecture

The internal architecture of the attention and feedforward modules are shown in the fig below:

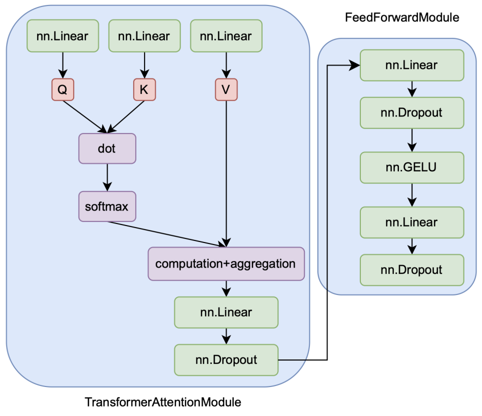

Figure 2: Attention and Feedforward Modules

A.3 FSGNN

The Features Selection Graph Neural Network (FSGNN) model takes a different approach. It feeds all computed features to the model with a weight parameter for each feature with the hope that the model itself will be able to learn which features are important. The architecture of the FSGNN model is shown in the figure below.

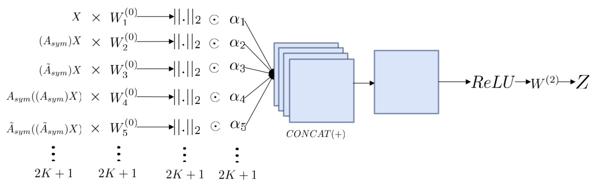
$X \times W_1^{(0)} \rightarrow ||.||_2 \odot \alpha_1$
$(A_{sym})X \times W_2^{(0)} \rightarrow ||.||_2 \odot \alpha_2$
$(\tilde{A}_{sym})X \times W_3^{(0)} \rightarrow ||.||_2 \odot \alpha_3$
$A_{sym}((A_{sym})X) \times W_4^{(0)} \rightarrow ||.||_2 \odot \alpha_4$
$\tilde{A}_{sym}((\tilde{A}_{sym})X) \times W_5^{(0)} \rightarrow ||.||_2 \odot \alpha_5$
$2K+1$
$2K+1$
$2K+1$
$2K+1$
$CONCAT(+)$
$ReLU \rightarrow W^{(2)} \rightarrow Z$

Figure 3: FSGNN Architecture

In the figure,

$$A_{sym} = D^{\frac{-1}{2}} A D^{\frac{-1}{2}}$$
$$\tilde{A}_{sym} = \tilde{D}^{\frac{-1}{2}} \tilde{A} \tilde{D}^{\frac{-1}{2}}$$

where $\tilde{A}$ is the adjacency matrix with added self loops and $\tilde{D}$ is its corresponding degree matrix.

Features of nodes K hops away are all computed in advance and are then multiplied by their corresponding weight matrix $W_k^{(0)}$. The result is then normalized and the hadamard product is then computed with a scalar weight α_k. These matrices are then concatenated via elementwise addition, passed through a ReLU layer and then multiplied with another weight matrix $W^{(2)}$ to get the final embeddings.

The architecture's basic assumption is that the model should be able to learn which features are important and α would be adjust accordingly.