Pitfalls of Graph Neural Network Evaluation

Oleksandr Shchur[*], **Maximilian Mumme**[*], **Aleksandar Bojchevski, Stephan Günnemann**
Technical University of Munich, Germany
{shchur,mumme,a.bojchevski,guennemann}@in.tum.de

Abstract

Semi-supervised node classification in graphs is a fundamental problem in graph mining, and the recently proposed graph neural networks (GNNs) have achieved unparalleled results on this task. Due to their massive success, GNNs have attracted a lot of attention, and many novel architectures have been put forward. In this paper we show that existing evaluation strategies for GNN models have serious shortcomings. We show that using the same train/validation/test splits of the same datasets, as well as making significant changes to the training procedure (e.g. early stopping criteria) precludes a fair comparison of different architectures. We perform a thorough empirical evaluation of four prominent GNN models and show that considering different splits of the data leads to dramatically different rankings of models. Even more importantly, our findings suggest that simpler GNN architectures are able to outperform the more sophisticated ones if the hyperparameters and the training procedure are tuned fairly for all models.

1 Introduction

Semi-supervised node classification in graphs is a classic problem in graph mining with applications ranging from e-commerce to computational biology. The recently proposed graph neural network architectures have achieved unprecedented results on this task and significantly advanced the state of the art. Despite their massive success, we cannot accurately judge the progress being made due to certain problematic aspects of the empirical evaluation procedures. We can partially attribute this to the practice of replicating the experimental settings from earlier works, since they are perceived as standard. First, a number of proposed models have all been tested exclusively on the same train/validation/test splits of the same three datasets (CORA, CiteSeer and PubMed) from Yang et al. [2016]. Such experimental setup favors the model that overfits the most and defeats the main purpose of using a train/validation/test split — finding the model with the best generalization properties [Friedman et al., 2001]. Second, when evaluating performance of a new model, people often use a training procedure that is rather different from the one used for the baselines. This makes it difficult to identify whether the improved performance comes from (a) a superior architecture of the new model, or (b) a better-tuned training procedure and / or hyperparameter configuration that unfairly benefits the new model [Lipton and Steinhardt, 2018].

In this paper we address these issues and perform a thorough experimental evaluation of four prominent GNN architectures on the transductive semi-supervised node classification task. We implement the four models – GCN [Kipf and Welling, 2017], MoNet [Monti et al., 2017], GraphSage [Hamilton et al., 2017] and GAT [Velickovic et al., 2018] – within the same framework.[1] In our evaluation we focus on two aspects: We use a standardized training and hyperparameter selection procedure for all models. In such a setting, the differences in performance can with high certainty be attributed to the differences in model architectures, not other factors. Second, we perform experiments

[*]Equal contribution
[1]Code is available at https://www.kdd.in.tum.de/gnn-benchmark

Relational Representation Learning Workshop (R2L 2018), NeurIPS 2018, Montréal, Canada.

on four well-known citation network datasets, as well as introduce four new datasets for the node classification problem. For each dataset we use 100 random train/validation/test splits and perform 20 random initializations for each split. This setup allows us to more accurately assess the generalization performance of different models, and does not just select the model that overfits one fixed test set.

Before we continue, we would like to make a disclaimer, that we do not believe that accuracy on benchmark datasets is the only important characteristic of a machine learning algorithm. Developing and generalizing the theory for existing methods, establishing connections to (and adapting ideas from) other fields are important research directions that move the field forward. However, thorough empirical evaluation is crucial for understanding the strengths and limitations of different models.

2 Models

We consider the problem of semi-supervised transductive node classification in a graph, as defined in Yang et al. [2016]. In this paper we compare the four following popular graph neural network architectures. **Graph Convolutional Network (GCN)** [Kipf and Welling, 2017] is one of the earlier models that works by performing a linear approximation to spectral graph convolutions. **Mixture Model Network (MoNet)** [Monti et al., 2017] generalizes the GCN architecture and allows to learn adaptive convolution filters. The authors of **Graph Attention Network (GAT)** [Velickovic et al., 2018] propose an attention mechanism that allows to weigh nodes in the neighborhood differently during the aggregation step. Lastly, **GraphSAGE** [Hamilton et al., 2017] focuses on inductive node classification, but can also be applied for transductive setting. We consider 3 variants of the GraphSAGE model from the original paper, denoted as **GS-mean**, **GS-meanpool** and **GS-maxpool**.

The original papers and reference implementations of all above-mentioned models consider different training procedures including different early stopping strategies, learning rate decay, full-batch vs. mini-batch training (a more detailed description is provided in Appendix A). Such diverse experimental setups makes it hard to empirically identify the driver behind the improved performance [Lipton and Steinhardt, 2018]. Thus, in our experiments we use a standardized training and hyperparameter tuning procedure for all models (more details in Sec. 3) to perform a more fair comparison.

In addition, we consider four baseline models. **Logistic Regression (LogReg)** and **Multilayer Perceptron (MLP)** are attribute-based models that do not consider the graph structure. **Label Propagation (LabelProp)** and **Normalized Laplacian Label Propagation (LabelProp NL)** [Chapelle et al., 2009], on the other hand, only consider the graph structure and ignore the node attributes.

3 Evaluation

Datasets For our experiments, we used the four well-known citation network datasets: PubMed [Namata et al., 2012], CiteSeer and CORA from Sen et al. [2008], as well as the extended version of CORA from Bojchevski and Günnemann [2018], denoted as CORA-Full. We also introduce four new datasets for the node classification task: Coauthor CS, Coauthor Physics, Amazon Computers and Amazon Photo. Descriptions of these new datasets, as well as statistics for all datasets can be found in Appendix B. For all datasests, we treat the graphs as undirected and only consider the largest connected component.

Setup We keep the *model architectures* as they are in the original papers / reference implementations. This includes the type and sequence of layers, choice of activation functions, placement of dropout, and choices as to where to apply L_2 regularization. We also fixed the number of attention heads for GAT to 8 and the number of Gaussian kernels for MoNet to 2, as proposed in the respective papers. All the models have 2 layers (input features $\rightarrow$ hidden layer $\rightarrow$ output layer).

For a more balanced comparison, however, we use the same *training procedure* for all the models. That is, we used the same optimizer (Adam [Kingma and Ba, 2015] with default parameters), same initialization (weights initialized according to Glorot and Bengio [2010], biases initialized with zeros), no learning rate decay, same maximum number of training epochs, early stopping criterion, patience and validation frequency (display step) for all models (Appendix C). We optimize all model parameters (attention weights for GAT, kernel parameters for MoNet, weight matrices for all models) simultaneously. In all cases we use full-batch training (using all nodes in the training set every epoch).

	CORA	CiteSeer	PubMed	CORA Full
GCN	81.5 ± 1.3	**71.9** ± 1.9	77.8 ± 2.9	**62.2** ± 0.6
GAT	**81.8** ± 1.3	71.4 ± 1.9	**78.7** ± 2.3	51.9 ± 1.5
MoNet	81.3 ± 1.3	71.2 ± 2.0	78.6 ± 2.3	59.8 ± 0.8
GS-mean	79.2 ± 7.7	71.6 ± 1.9	77.4 ± 2.2	58.6 ± 1.6
GS-maxpool	76.6 ± 1.9	67.5 ± 2.3	76.1 ± 2.3	40.7 ± 1.5
GS-meanpool	77.9 ± 2.4	68.6 ± 2.4	76.5 ± 2.4	40.5 ± 1.5
MLP	58.2 ± 2.1	59.1 ± 2.3	70.0 ± 2.1	36.8 ± 1.0
LogReg	57.1 ± 2.3	61.0 ± 2.2	64.1 ± 3.1	40.5 ± 0.8
LabelProp	74.4 ± 2.6	67.8 ± 2.1	70.5 ± 5.3	50.5 ± 1.5
LabelProp NL	73.9 ± 1.6	66.7 ± 2.2	72.3 ± 2.9	51.0 ± 1.0
	Coauthor CS	**Coauthor Physics**	**Amazon Computer**	**Amazon Photo**
GCN	91.1 ± 0.5	92.8 ± 1.0	82.6 ± 2.4	91.2 ± 1.2
GAT	90.5 ± 0.6	92.5 ± 0.9	78.0 ± 19.0	85.7 ± 20.3
MoNet	90.8 ± 0.6	92.5 ± 0.9	**83.5** ± 2.2	91.2 ± 1.3
GS-mean	**91.3** ± 2.8	**93.0** ± 0.8	82.4 ± 1.8	**91.4** ± 1.3
GS-maxpool	85.0 ± 1.1	90.3 ± 1.2	N/A	90.4 ± 1.3
GS-meanpool	89.6 ± 0.9	92.6 ± 1.0	79.9 ± 2.3	90.7 ± 1.6
MLP	88.3 ± 0.7	88.9 ± 1.1	44.9 ± 5.8	69.6 ± 3.8
LogReg	86.4 ± 0.9	86.7 ± 1.5	64.1 ± 5.7	73.0 ± 6.5
LabelProp	73.6 ± 3.9	86.6 ± 2.0	70.8 ± 8.1	72.6 ± 11.1
LabelProp NL	76.7 ± 1.4	86.8 ± 1.4	75.0 ± 2.9	83.9 ± 2.7

Table 1: Mean test set accuracy and standard deviation in percent averaged over 100 random train/validation/test splits with 20 random weight initializations each for all models and all datasets. For each dataset, the highest accuracy score is marked in **bold**. N/A stands for the dataset that couldn't be processed by the full-batch version of GS-maxpool because of GPU RAM limitations.

Lastly, we used the exact same strategy for *hyperparameter selection* for every model. We performed an extensive grid search for learning rate, size of the hidden layer, strength of the L_2 regularization, and dropout probability (Appendix C). We restricted the random search space to ensure that every model has at most the same given number of trainable parameters. For every model, we picked the hyperparameter configuration that achieved the best average accuracy on Cora and CiteSeer datasets (averaged over 100 train/validation/test splits and 20 random initializations for each). The chosen best-performing configurations were used for all subsequent experiments and are listed in Table 4. In all cases, we use 20 labeled nodes per class as the training set, 30 nodes per class as the validation set, and the rest as the test set.

Results Table 1 shows mean accuracies (and their standard deviations[2]) of all models for all 8 datasets averaged over 100 splits and 20 random initializations for each split. There are a few observations to be made. First, the GNN-based approaches (GCN, MoNet, GAT, GraphSAGE) significantly outperform all the baselines (MLP, LogReg, LabelProp, LabelProp NL) across all the datasets. This matches our intuition and confirms the superiority of GNN-based approaches that combine both the structural and attribute information compared to methods considering only the attributes or only the structure.

Among the GNN approaches, there is no clear winner that dominates across all the datasets. In fact, for 5 out of 8 datasets, scores of the 2nd and 3rd best approaches are less than 1% away from the average score of the best-performing method. If we were interested in comparing one model versus the rest, we could perform pairwise t-tests, as done in Klicpera et al. [2019]. Since we are interested in comparing all the models to each other, we consider the relative accuracy of each model instead. For this, we take the best accuracy score for each split of each dataset (already averaged

[2]Standard deviations are not the best representation of the variance of the accuracy scores, since the scores are not normally distributed. We still include the standard deviations to give the reader a rough idea of the variance of the results for each model. A more accurate picture is given by the box plots in Figure 1.

	Relative accuracy	Avg. rank
GCN	99.4	2.3
MoNet	99.0	2.7
GS-mean	98.3	2.7
GAT	95.9	3.6
GS-meanpool	93.0	5.2
GS-maxpool	91.1	6.4
LabelProp NL	89.3	7.4
LabelProp	86.6	7.7
LogReg	80.6	8.8
MLP	77.8	8.8

(a) Relative accuracy and average rank.

Planetoid split	**CORA**	**CiteSeer**	**PubMed**
GCN	81.9 ± 0.8	69.5 ± 0.9	$\mathbf{79.0} \pm 0.5$
GAT	$\mathbf{82.8} \pm 0.5$	$\mathbf{71.0} \pm 0.6$	77.0 ± 1.3
MoNet	82.2 ± 0.7	70.0 ± 0.6	77.7 ± 0.6
GS-maxpool	77.4 ± 1.0	67.0 ± 1.0	76.6 ± 0.8
Another split	**CORA**	**CiteSeer**	**PubMed**
GCN	$\mathbf{79.0} \pm 0.7$	$\mathbf{68.6} \pm 1.1$	69.5 ± 1.0
GAT	77.9 ± 0.7	67.7 ± 1.2	69.5 ± 0.6
MoNet	77.9 ± 0.7	66.8 ± 1.3	$\mathbf{70.7} \pm 0.5$
GS-maxpool	74.5 ± 0.6	63.1 ± 1.2	70.3 ± 0.8

(b) Different split leads to a completely different ranking of models.

Table 2: (a) Relative accuracy scores and ranks averaged over all datasets. See text for the definition. (b) Model accuracy on the Planetoid split from Yang et al. [2016] and another split on the same datasets. Different splits lead to a completely different ranking of models.

over 20 initializations) as 100%. Then, the score of each model is divided by this number, and the results for each model are averaged over all the datasets and splits. We also rank algorithms by their performance (1 = best performance, 10 = worst), and compute the average rank across all datasets and splits for each algorithm. The final scores are reported in Table 2a. We observe that GCN is able to achieve the best performance across all models. While this result seems surprising, similar findings have been reported in other fields. Simpler models often outperform more sophisticated ones if hyperparameter tuning is performed equally carefully for all methods [Melis et al., 2018, Lucic et al., 2017]. In future work, we plan to further investigate what are the specific properties of the graphs that lead to the differences in performance of the GNN models.

Another surprising finding is the relatively lower score and high variance in results obtained by GAT for the Amazon Computers and Amazon Photo datasets. To investigate this phenomenon, we additionally visualize the accuracy scores achieved by different models on the Amazon Photo dataset in Figure 2 in the appendix. While the median scores for all GNN models are very close to each other, GAT produces extremely low scores (below 40%) for some weight initializations. While these outliers occur rarely (for 138 out of 2000 runs), they significantly lower the average score of GAT.

Effect of the train/validation/test split To demonstrate the effect of different train/validation/test splits on the performance, we execute the following simple experiment. We run the 4 models on the datasets and respective splits from [Yang et al., 2016]. As shown in Table 2b, GAT achieves the best scores for the CORA and CiteSeer datasets, and GCN gets the top score for PubMed. If we, however, consider a different random split with the same train/validation/test set sizes the ranking of models is completely different, with GCN being first on CORA and CiteSeer, and MoNet winning on PubMed. This shows how fragile and misleading results obtained on a single split can be. Taking further into account that the predictions of GNNs can greatly change under small data perturbations [Zügner et al., 2018] clearly confirms the need for evaluation strategies based on multiple splits.

4 Conclusion

We have performed an empirical evaluation of four state-of-the-art GNN architectures on the node classification task. We introduced four new attributed graph datasets, as well as open-sourced a framework that enables a fair and reproducible comparison of different GNN models. Our results highlight the fragility of experimental setups that consider only a single train/validation/test split of the data. We also find that, surprisingly, a simple GCN model can outperform the more sophisticated GNN architectures if the same hyperparameter selection and training procedures are used, and the results are averaged over multiple data splits. We hope that these results will encourage future works to use more robust evaluation procedures.

Acknowledgments

This research was supported by the German Research Foundation, Emmy Noether grant GU 1409/2-1.

References

A. Bojchevski and S. Günnemann. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *ICLR*, 2018.

O. Chapelle, B. Scholkopf, and A. Zien. *Semi-supervised learning*. 2009.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. 2001.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *NIPS*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. *ICLR*, 2019.

Z. C. Lipton and J. Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? A large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.

G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. *ICLR*, 2018.

F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. *CVPR*, 2017.

G. Namata, B. London, L. Getoor, and B. Huang. Query-driven active surveying for collective classification. 2012.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29, 2008.

P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. *ICLR*, 2018.

Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *ICML*, 2016.

D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*, 2018.

A Differences in training procedures for GNN models

GCN

- Early stopping: stop optimization if the validation loss is larger than the mean of validation losses of the last 10 epochs.
- Full-batch training.
- Maximum number of epochs: 200.
- Train set: 20 per class; validation set: 500 nodes; test set: 1000 (as in the Planetoid split).

MoNet

- No early stopping.
- Full-batch training.
- Maximum number of epochs: 3000 for CORA, 1000 for PubMed.
- Train set: 20 per class; validation set: 500 nodes; test set: 1000 (as in the Planetoid split).
- Alternating optimization of weight matrices and kernel parameters.
- Learning rate decay at predefined iterations (only for CORA).

GAT

- Early stopping: stop optimization if neither the validation loss nor the validation accuracy improve for 100 epochs.
- Full-batch training.
- Maximum number of epochs: 100000.
- Train set: 20 per class; validation set: 500 nodes; test set: 1000 (as in the Planetoid split).

GraphSAGE

- No early stopping.
- Mini-batch training with batch size of 512.
- Maximum number of epochs (each epoch consists of multiple mini-batches): 10.

B Datasets description and statistics

Amazon Computers and Amazon Photo are segments of the Amazon co-purchase graph [McAuley et al., 2015], where nodes represent goods, edges indicate that two goods are frequently bought together, node features are bag-of-words encoded product reviews, and class labels are given by the product category.

Coauthor CS and Coauthor Physics are co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge [3]. Here, nodes are authors, that are connected by an edge if they co-authored a paper; node features represent paper keywords for each author's papers, and class labels indicate most active fields of study for each author.

	Classes	**Features**	**Nodes**	**Edges**	**Label rate**	**Edge density**
CORA	7	1433	2485	5069	0.0563	0.0004
CiteSeer	6	3703	2110	3668	0.0569	0.0004
PubMed	3	500	19717	44324	0.0030	0.0001
CORA-Full	67	8710	18703	62421	0.0745	0.0001
Coauthor CS	15	6805	18333	81894	0.0164	0.0001
Coauthor Physics	5	8415	34493	247962	0.0029	0.0001
Amazon Computers	10	767	13381	245778	0.0149	0.0007
Amazon Photo	8	745	7487	119043	0.0214	0.0011

Table 3: Dataset statistics after standardizing the graphs, adding self-loops and removing classes with too few instances from CORA_full. We ignore 3 classes with less than 50 nodes in CORA-Full dataset (since we cannot perform the 20/30/rest split for them).

Label rate is the fraction of nodes in the training set. Since we use 20 training instances per class this can be computed as $(\#\text{classes} \cdot 20)\ /\ \#\text{nodes}$.

The **edge density** describes the fraction of all possible edges that is present in the graph and can be computed as $\#\text{edges}\ /\ (\frac{1}{2} \cdot \#\text{nodes}^2)$.

[3] `https://kddcup2016.azurewebsites.net/`

C Hyperparameter configurations and Early Stopping

Grid search was performed over the following search space:

- Hidden size: `[8, 16, 32, 64]`
- Learning rate: `[0.001, 0.003, 0.005, 0.008, 0.01]`
- Dropout probability: `[0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]`
- Attention coefficients dropout probability (only for GAT): `[0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]`
- L_2 regularization strength: `[1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 5e-2, 1e-1]`

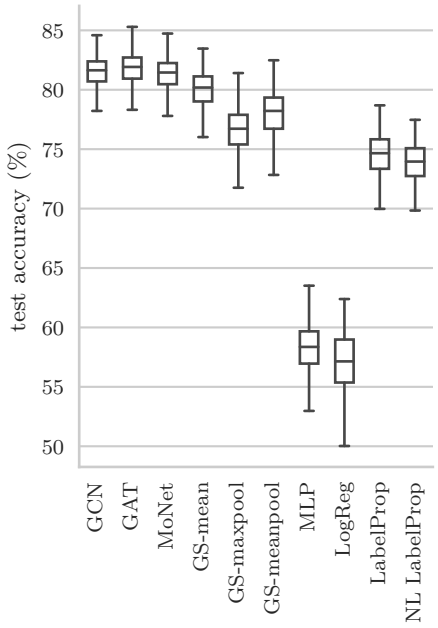
We train for a maximum of 100k epochs. However, the actual training time is considerably shorter since we use strict early stopping. Specifically, with our unified early stopping criterion training stops if the total validation loss (loss on the data plus regularization loss) does not improve for 50 epochs. Once training has stopped, we reset the state of the weights to the step with the lowest validation loss.

	Effective hidden size	**Learning rate**	**Dropout**	L_2 **reg. strength**	**Trainable weights**
GCN	64	0.01	0.8	0.001	$92K$
GAT	64	0.01	0.6/0.3	0.01	$92K$
MoNet	64	0.003	0.7	0.05	$92K$
GS-mean	32	0.001	0.4	0.1	$92K$
GS-maxpool	32/32	0.001	0.3	0.005	$94K$
GS-meanpool	32/8	0.001	0.2	0.01	$58K$
MLP	64	0.005	0.8	0.01	$92K$
LogReg	–	0.1	–	0.0005	$10K$

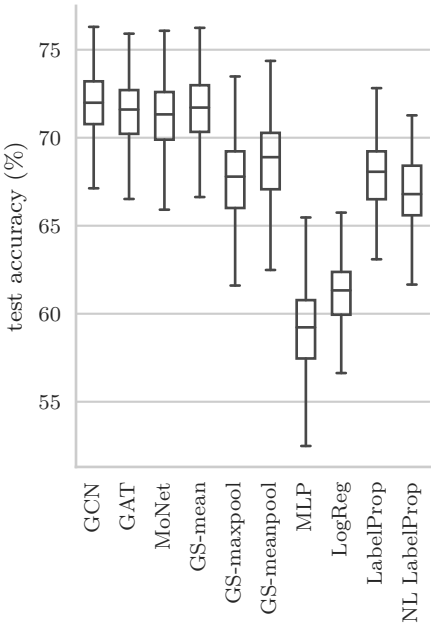
Table 4: Best performing hyperparameter configurations for each model chosen by grid search. GAT has two dropout probabilities (dropout on features / dropout on attention coefficients). All GraphSAGE models have additional weights for the skip connections (which effectively doubles the hidden size). GS-meanpool/GS-maxpool have two hidden sizes (hidden layer size / size of intermediary feature transformation). GAT uses a multi-head architecture with 8 heads and MoNet uses 2 heads, so the hidden state is split over 8 and 2 heads respectively.

D Performance of different models across datasets

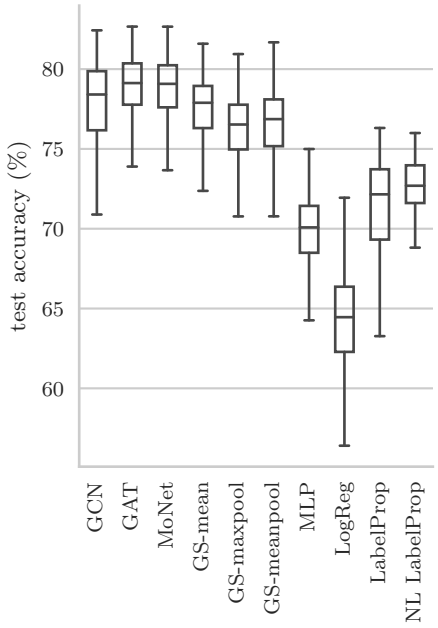
(a) **CORA**



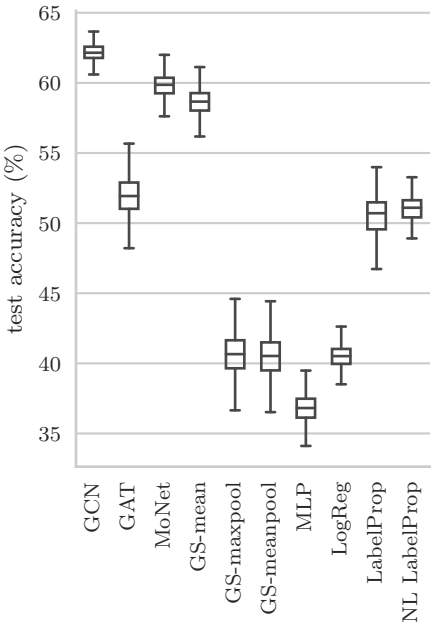
(b) **CiteSeer**



(c) **PubMed**



(d) **CORA-Full**



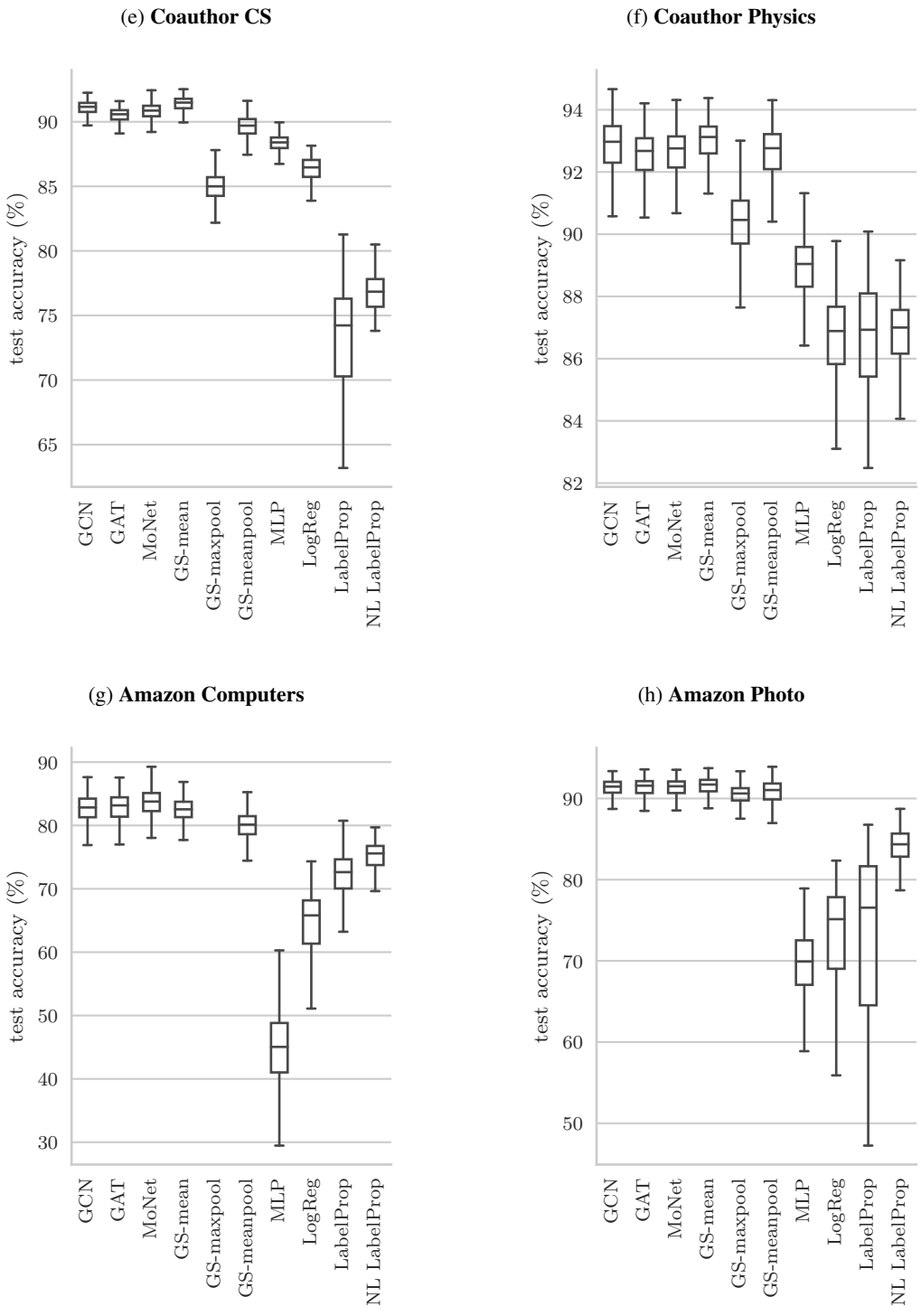


Figure 1: Boxplots of the test set accuracy of all models on all datasets over 100 random train/validation/test splits with 20 random weight initializations each. Note that a boxplot displays the median of the data as well as the 50% quantiles. Note further that outliers are excluded in these plots since some models have outliers very far from the median which would shrink the resolution of the plots. For a plot including the outliers refer to Figure 2.

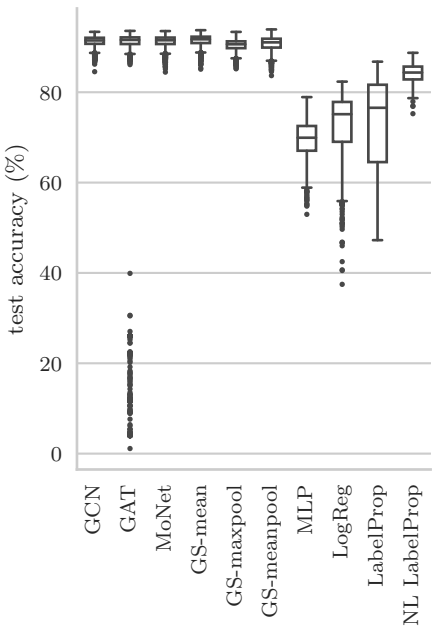


Figure 2: Boxplot showing outliers for the **Amazon Photo** dataset.