Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks

Peter Mernyei [1] **Cătălina Cangea** [1]

Abstract

We present WIKI-CS, a novel dataset derived from Wikipedia for benchmarking Graph Neural Networks. The dataset consists of nodes corresponding to Computer Science articles, with edges based on hyperlinks and 10 classes representing different branches of the field. We use the dataset to evaluate semi-supervised node classification and single-relation link prediction models. Our experiments show that these methods perform well on a new domain, with structural properties different from earlier benchmarks. The dataset is publicly available, along with the implementation of the data pipeline and the benchmark experiments, at `https://github.com/pmernyei/wiki-cs-dataset`.

1. Introduction

In recent years, significant advances have been made in learning representations of graph-structured data and predicting quantities of interest for nodes, edges or graphs themselves (Kipf & Welling, 2016a; Veličković et al., 2017). This new subfield has attracted an increasing amount of interest, leading to the development of numerous methods (Wu et al., 2019). However, several earlier works have noted issues with existing standard benchmarks, which make it difficult to rigorously compare results and accurately distinguish between the performance of competing architectures (Shchur et al., 2018; Klicpera et al., 2018).

Our primary focus is semi-supervised node classification: given labels of a small subset of nodes (typically 1–5%) and features of all nodes, as well as their connectivity information, the task is to predict all other labels. This setup is often used to assess the performance of various Graph Neural Network (GNN) architectures (Kipf & Welling, 2016a; Veličković et al., 2017). These methods are usually evaluated on three citation network datasets (Cora, CiteSeer, PubMed) introduced by Yang et al (2016). Unfortunately, different training splits are used across studies, which makes comparisons challenging, especially since the performance on this task is very sensitive to the choice of training split (Shchur et al., 2018). Furthermore, all three benchmarks are derived from the same domain, with similar structural properties. In contrast, WIKI-CS has a much higher connectivity rate, hence it provides a different kind of distribution for these methods to be tested on.

Similar datasets have also been used in single-relation link prediction (Kipf & Welling, 2016b; Haonan et al., 2019). We further use WIKI-CS to benchmark relational methods for this task, along with a non-structural SVM baseline.

2. Related Work

The most commonly used semi-supervised node classification benchmarks are the previously-described citation network graphs, proposed by Yang et al. (2016). Larger datasets have also been used, such as Reddit and PPI (Hamilton et al., 2017). However, due to the standard split sizes proposed for these benchmarks, state-of-the-art methods have already achieved F1 scores of 0.995 and 0.97 respectively (Zeng et al., 2019), making it difficult for further improvements to be properly gauged.

Due to the issues with existing datasets, there has been significant concurrent work on establishing robust GNN benchmarks:

- The Open Graph Benchmark (Hu et al., 2020) (OGB) has recently developed a range of datasets, focusing on diversity of domains, graph sizes and types of tasks and unified evaluation methods. A Wikidata Knowledge Graph is included for a link prediction task—note that this source material is entirely different from the article hyperlink graph used for WIKI-CS. OGB also proposes challenging domain-specific splits based on some aspect of the data (for exaxmple, time or molecular structure), instead of selecting this randomly.

- Dwivedi et al. (2020) similarly proposed several datasets to rigorously distinguish the aspects of GNN architectures that significantly contribute to good per-

[1]Department of Computer Science, University of Cambridge, Cambridge, United Kingdom. Correspondence to: Peter Mernyei <pmernyei@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

formance on challenging benchmarks. To achieve this, they used largely synthetic graphs.

Our contribution complements the existing ones by providing a dataset and experimental results based on a new domain. We thus further establish the generality of GNN methods and extend the range of available benchmarks.

3. The Dataset

3.1. Article Selection and Label Generation

We processed Wikipedia datadumps from August 2019 to extract a subgraph where accurate class labels could be provided, based on the categories of each article. Unfortunately, these category tags were not suitable for direct use as class labels, as most of them are highly specific and inconsistently applied to a small number of pages—there were around 1.5 million different categories defined on the 6 million pages, at the time of the snapshot that we used.

This problem was mitigated by using the category sanitizer tool made available by Boldi & Monti (2016), with some modifications. Their method relies on the subcategory relation to aggregate articles belonging to subcategories to their parent categories. A small set of prominent categories is selected based on harmonic centrality measures in the subcategory graph; other nodes in the subcategory graph are aggregated to one of their nearest ancestors (see Figure 1 for an example subgraph). See Boldi & Monti (2016) for the details of the process. This avoids aggregation through many indirect steps, which would often lead to articles being mapped to categories which they have little semantic overlap with.

However, the output still required further clean-up: some aggregated categories still contained unrelated articles. Additionally, if the subcategory graph related to some topic is very dense, the selected prominent categories and the aggregation choices can be very arbitrary.

WIKI-CS was created by inspecting the list of $10,000$ prominent categories selected by the sanitizer and picking a subject area with few such issues. We identified three possible candidate subjects (branches of biology, US states, branches of CS), and sampled 20 pages from every class of these candidates. Although all of these had some issues, we were able to clean up the CS data by dropping some categories and manually disabling aggregation across specific subcategories to prune bad pages from others. This resulted in a dataset with 10 classes corresponding to branches of computer science, with very high connectivity. See Appendix B for the set of prominent categories we used for each label. Finally, we dropped the articles that would have been mapped to multiple classes.

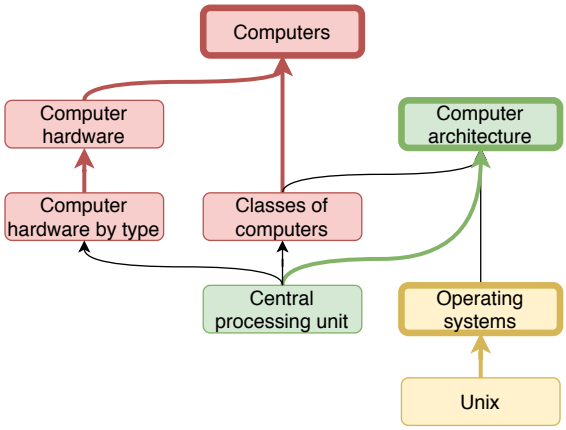


Figure 1. A subgraph of the subcategory relation graph. Nodes with dark borders are the prominent categories chosen based on centrality. The others were aggregated to the nearest marked ancestor as denoted by their colors, with ties broken arbitrarily.

3.2. Node Features

Similarly to previous work (Yang et al., 2016), our node features were derived from the text of the corresponding articles. However, they were calculated as the average of pre-trained GloVe word embeddings (Pennington et al., 2014) instead of using binary bag-of-words vectors. This allowed us to encode rich features corresponding to a large vocabulary in relatively small 300-dimensional input vectors, which can be an advantage for training large models on a GPU.

3.3. Training Splits

It has been shown that the choice of the training split can seriously affect model performance for semi-supervised node classification (Shchur et al., 2018). Therefore, using multiple training splits can improve the robustness of a benchmark (Klicpera et al., 2018). For this reason, we randomly selected 20 different training splits from the data that was not used for testing.

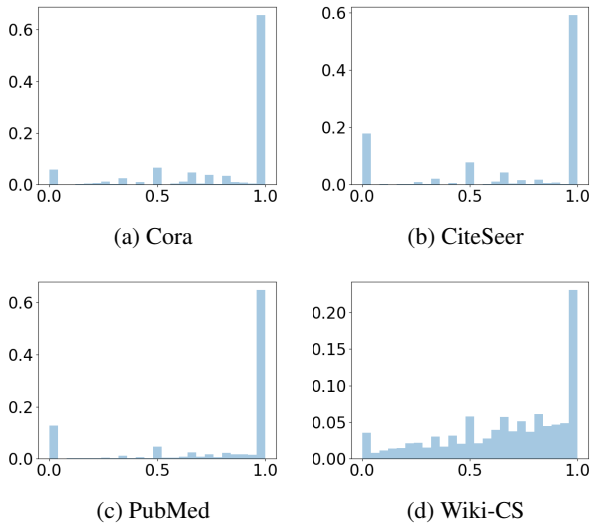
More specifically, we split the nodes in each class into two sets, 50% for the test set and 50% potentially visible. From the visible set, we generated 20 different splits of training, validation and early-stopping sets: 5% of the nodes in each class were used for training in each split, 22.5% were used to evaluate the early-stopping criterion, and 22.5% were used as the validation set for hyperparameter tuning. We stored the resulting mask vectors with the rest of the dataset, so that they can be used consistently across all future work.

3.4. Statistics and Structural Properties

Table 1 summarises the key statistics of the citation network and the WIKI-CS datasets. Note the significantly higher rate of connectivity compared to existing benchmarks and

Table 1. Comparison of key dataset statistics between WIKI-CS and standard citation network benchmarks. SP stands for shortest path length.

	Cora	CiteSeer	PubMed	Wiki-CS
Classes	7	6	3	10
Nodes	2708	3327	19717	11701
Edges	5429	4732	44338	216123
Features dim.	1433	3703	500	300
Label rate	3.6%	5.2%	0.3%	5%
Mean degree	4.00	2.84	4.50	36.94
Average SP	6.31	9.32	6.34	3.01



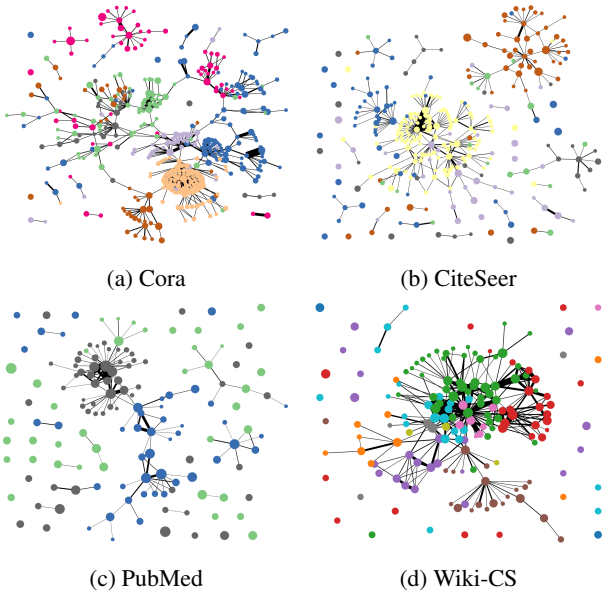
(a) Cora (b) CiteSeer (c) PubMed (d) Wiki-CS

Figure 2. Distribution of the ratio of neighbors belonging to the same class. In all three the citation network datasets, almost two-thirds of all nodes have all neighbors belonging to the same class. The distribution of WIKI-CS is considerably more balanced.

the short average distance between any two nodes. This suggests that progress could be made on the benchmark by designing more involved computations within the neighborhood of a node, rather than focusing on long-range connections. This makes WIKI-CS a useful and complementary addition to existing node classification datasets.

This connectivity also leads to more varied node neighborhoods: for each node, we calculated the proportion of neighbors that belong to the same class as the node itself, and plotted this distribution for WIKI-CS as well as the existing citation network benchmarks. The results shown in Figure 2 show that the existing datasets have a large share of nodes in homogeneous neighborhoods, while WIKI-CS is significantly more varied.

We also visualized the structure of all four datasets using Deep Graph Mapper (Bodnar et al., 2020), an unsupervised



(a) Cora (b) CiteSeer (c) PubMed (d) Wiki-CS

Figure 3. Deep Graph Mapper (DGM) visualisation of benchmarks. Each node in the figure corresponds to a cluster of similar nodes in the original graph, with edge thickness representing the amount of connections between clusters. Colors represent the most frequent class in each cluster. The DGM unsupervised embedding process did not take labels into account, only relying on the node features and edges. The hyperparameters are described in Appendix A.

GNN-based visualisation technique. The results shown in Figure 3 suggest that WIKI-CS might have a more centralized, hierarchical structure than the citation networks, which seems plausible considering the different source domains.

4. Experiments

4.1. Semi-Supervised Node Classification

As described in Section 3.3, 20 different training splits were created for the node classification task, each consisting of 5% of nodes from each class. The same test set (50% of the nodes) was evaluated for all splits. In each split, a different 22.5% of nodes is used for early-stopping: we finish training when the loss calculated on this set has not improved for 100 epochs, and evaluate the model snapshot that produced the lowest loss.

This evaluation was performed 5 times on each of the 20 splits; we report the mean accuracy with a 95% confidence interval based on bootstrap resampling from these results with 1,000 samples.

Three GNN models were evaluated: GCN (Kipf & Welling, 2016a), GAT (Veličković et al., 2017) and APPNP (Klicpera et al., 2018). Hyperparameter tuning was performed using the same training setup and measuring validation performance on 22.5% of the nodes disjoint from the training and

early-stopping sets. For efficiency, only the first 10 (out of 20) splits were used for hyperparameter tuning. The model configurations are described in Appendix A.

Two non-structural baselines were also included: a multi-layer perceptron (MLP) and a support vector machine (SVM). These predicted the class for each node individually, based on the node features. Since SVMs are deterministic, we only had a single data point from each training split and report the mean accuracy.

The results are shown in Table 2. The relative model performances align well with the results on citation network benchmarks, providing evidence that these are indeed good general-purpose methods. It is perhaps surprising that the attention mechanism of GAT improved very little on the GCN result despite the large neighborhoods—one reason might be that it is difficult to learn what to attend to in the semi-supervised setting, as discussed in-depth by Knyazev et al. (2019).

The model predictions were also visualised with Deep Graph Mapper, and are included in Appendix C. This was based on training each model once, on the first of the 20 training splits. As expected, the mistakes and disagreements are largely located near boundaries of classes. This reinforces the idea that more complex neighborhood aggregation methods might be able to improve prediction accuracy. There are also some less connected clusters that seem to produce consistent incorrect predictions under all models—this might be due to not having good training samples in their proximity.

Table 2. Performance of semi-supervised node classification methods on the WIKI-CS dataset. Accuracies are represented as the average over 100 runs, with 95% confidence intervals calculated by bootstrapping.

	ACCURACY
SVM	72.63%
MLP	$73.17 \pm 0.19\%$
GCN	$77.19 \pm 0.12\%$
GAT	$77.65 \pm 0.11\%$
APPNP	$79.84 \pm 0.10\%$

4.2. Link Prediction

For the link prediction benchmark, we followed the experimental setup of studies performing single-relation link prediction on the Cora, CiteSeer and PubMed datasets (Kipf & Welling, 2016b; Haonan et al., 2019). We split the data as follows: 85% of the real edges for training, 5% for validation and 10% for testing. For each group, the same number of negative examples (non-edge node pairs) was sampled uniformly at random.

Two GNN methods were benchmarked for link prediction: GraphStar (Haonan et al., 2019) and VGAE (Kipf & Welling, 2016b). They were trained using the configurations reported in the original works, except for the hidden layer size of GraphStar: a maximum size of 256 would fit on the GPU. Details are included in Appendix A. An MLP baseline was also trained using concatenated pairs of node feature vectors.

Table 3. Performance of link prediction methods on the WIKI-CS dataset. Metrics are represented as the average over 50 runs of VGAE, 20 runs of the MLP and 10 runs of GraphStar, with 95% confidence intervals calculated by bootstrapping.

	ROC-AUC	AP
MLP	0.9785 ± 0.0001	0.9761 ± 0.0002
VGAE	0.9553 ± 0.0008	0.9608 ± 0.0007
GRAPHSTAR	0.9793 ± 0.0002	0.9896 ± 0.0001

Table 4. Performance of link prediction methods trained on only $10,000$ examples of each class.

	ROC-AUC	AP
MLP	0.9192 ± 0.0004	0.9119 ± 0.0006
VGAE	0.8546 ± 0.0024	0.8427 ± 0.0032
GRAPHSTAR	0.9577 ± 0.0006	0.9795 ± 0.0003

The results are shown in Table 3. Note the extremely high performance of all models, even the MLP baseline. It appears that randomly selected false edges are very easy to distinguish from true edges in this dataset, and harder negative samples would be needed for more meaningful benchmarking. The large number of edges aggravates this, but it is not the main cause: we performed an experiment where we trained the models on just 10000 examples of each class, and found the metrics to be still comfortably above 0.9. See Table 4 for the results.

5. Conclusion

We have presented WIKI-CS, a new benchmark for GNN methods. We have described how its structural properties are significantly different from commonly used datasets. Our experiments show existing GNN architectures for semi-supervised node classification and link prediction performing similarly to their results on other benchmarks, which is further evidence that they are good general-purpose methods for graph-learning tasks. Our dataset is available for further study, broadening the range of available benchmarks.

References

Bodnar, C., Cangea, C., and Liò, P. Deep graph mapper: Seeing graphs through the neural lens. *arXiv preprint arXiv:2002.03864*, 2020.

Boldi, P. and Monti, C. Cleansing Wikipedia Categories using Centrality. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pp. 969–974. International World Wide Web Conferences Steering Committee, 2016.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Haonan, L., Huang, S. H., Ye, T., and Xiuyan, G. Graph Star Net for Generalized Multi-Task Learning. *arXiv preprint arXiv:1906.12330*, 2019.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907, 2016a. URL http://arxiv.org/abs/1609.02907.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Klicpera, J., Bojchevski, A., and Günnemann, S. Personalized Embedding Propagation: Combining Neural Networks on Graphs with Personalized PageRank. *CoRR*, abs/1810.05997, 2018. URL http://arxiv.org/abs/1810.05997.

Knyazev, B., Taylor, G. W., and Amer, M. R. Understanding attention in graph neural networks. *arXiv preprint arXiv:1905.02850*, 2019.

Pennington, J., Socher, R., and Manning, C. D. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A Comprehensive Survey on Graph Neural Networks. *arXiv e-prints*, art. arXiv:1901.00596, Jan 2019.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

A. Hyperparameter settings

A.1. Node classification

The following hyperaparameters were used for the node classification models:

- **MLP:** 2 layers, 35 units in the hidden layer, 0.35 dropout probability, learning rate 0.003.
- **SVM:** radial basis function kernel with $C = 8$.
- **GCN:** 2 layers, 33 hidden neurons, learning rate 0.02, 0.25 dropout probability.
- **GAT:** 2 layers, 5 attention heads for the hidden layer outputting 14 units each, 0.5 dropout probability, learning rate 0.007.
- **APPNP:** learning rate 0.02, 0.4 dropout probability, propagation iterations $k = 2$, teleport probability $\alpha = 0.11$.

Additionally, all models used an L2 loss coefficient of 5×10^{-4}, and the evaluation was performed with self-loops added to all nodes.

A.2. Link prediction

- **GraphStar:** the original training code with a dynamically varying learning rate schedule was used, trained for 1,000 epochs and the test metrics reported from the epoch with the highest average validation metrics. A hidden layer size of 256 was used to fit into memory and all other parameters as given by the authors: 3 layers, no dropout, 5×10^{-4} L2 regularisation.
- **VGAE:** largely the same setup was used as the original paper: 16-dimensional latent space, GCN encoders with a shared hidden layer of 32 neurons, no dropout, trained for 200 epochs. However, we found improved performance when discarding the KL regularisation loss on the latent space distributions.
- **MLP:** a fully connected network with 3 hidden layers of 128 units each, trained for 100 epochs with 0.2 dropout probability after each layer, learning rate 10^{-4} and no weight decay.

A.3. DGM visualisations

All DGM plots in Figures 3 and 4 were created with the SDGM variant (which visualises the original graph structure), the unsupervised DGI lens, t-SNE reduction, 20 intervals, and the following additional parameters:

- **Cora:** $\epsilon = 0.05$, min component size 8.
- **CiteSeer:** $\epsilon = 0.03$, min component size 8.
- **PubMed:** $\epsilon = 0.25$, min component size 12.
- **Wiki-CS:** $\epsilon = 0.05$, min component size 10.

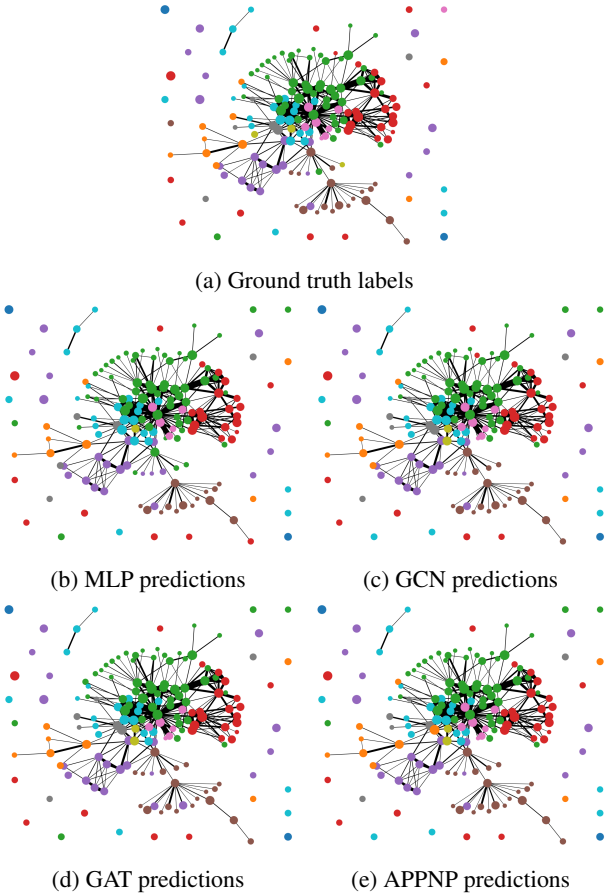
B. List of categories for each label

Table 5 shows the list of categories used to construct each class. The listed categories have been selected by the sanitizer tool as aggregation targets, so each class consists of pages that were aggregated to one of the corresponding targets. Some subcategories of these targets that added unsuitable results were manually excluded.

Table 5. List of aggregated Wikipedia categories used to construct each class.

ID	MAIN CATEGORIES
0	COMPUTATIONAL LINGUISTICS
1	DATABASES
2	OPERATING SYSTEMS OPERATING SYSTEMS TECHNOLOGY
3	COMPUTER ARCHITECTURE
4	COMPUTER SECURITY COMPUTER NETWORK SECURITY ACCESS CONTROL DATA SECURITY COMPUTATIONAL TRUST COMPUTER SECURITY EXPLOITS
5	INTERNET PROTOCOLS
6	COMPUTER FILE SYSTEMS
7	DISTRIBUTED COMPUTING ARCHITECTURE
8	WEB TECHNOLOGY WEB SOFTWARE WEB SERVICES
9	PROGRAMMING LANGUAGE TOPICS PROGRAMMING LANGUAGE THEORY PROGRAMMING LANGUAGE CONCEPTS PROGRAMMING LANGUAGE CLASSIFICATION

C. Deep Graph Mapper visualisation of model predictions



(a) Ground truth labels

(b) MLP predictions

(c) GCN predictions

(d) GAT predictions

(e) APPNP predictions

Figure 4. Deep Graph Mapper visualisation of the predictions of different node classification models. The top image colors each cluster according to its most frequent true label, similar to Figure 3d. The other plots have clusters colored according to the most frequent prediction of the appropriate model. Note that this can hide differences that do not change the majority prediction in a cluster. The specific parameters used are described in Appendix A.