Applications of Deep Learning on Graphs: Project 1

Yves Bicker
Universität Zurich
bickery@student.ethz.ch
16-924-128

Moritz Dück
ETH Zurich
mdueck@student.ethz.ch
21-950-761

1 Task 2: Transductive Learning

The Cora dataset consists of 2708 scientific publications represented as nodes, which are classified into seven classes (research topics). The number of edges is 10556. There are no isolated nodes, no self-loops. The edges are undirected which leads to an edge count that is twice the number of actual citation links. Each publication in the dataset is described by a binary feature vector of size 1433 indicating the absence or presence of each unique word. The longest shortest path in the Graph consists of 19 Edges. The node degree distribution shows that the majority of nodes has a node degree of 10 or less, while a few publications are hubs with a degree of up to 168. The distribution of the node labels shows an over proportion of label 3. See Appendix B

Label propagation with full observation: The majority voting classifier is allowed to observe all labels of the full graph. Classification for a node is based on the majority label of it's neighbours. If there is a tie, we sample the label uniformly at random. The high prediction accuracy of 0.8613 ± 0.0012 indicates that papers that cite each other belong to the same research topic.

Baseline without Graph Structure: The random forest classifier is used on the public train/val/test split, where hyper-parameter tuning is done on the number of trees. The classifier is trained on the node features only. The accuracy of 0.5770 ± 0.0050 suggests that the raw features provide moderate predictive power for classifying publications in the absence of citation information.

Task	Model	Accuracy			
		Seed			$\mu \pm \sigma$
		42	43	44	
2.2	Majority label classifiern	0.8610	0.8600	0.8630	0.8613 ± 0.0012
2.3	RF w/o graph structure	0.5880	0.5770	0.5850	0.5833 ± 0.0046
2.4	RF on GNN embeddings	0.5280	0.5220	0.5380	0.5293 ± 0.0066
	RF on MLP embeddings	0.3110	0.3060	0.2990	0.3053 ± 0.0049
	RF on Random features	0.1350	0.1470	0.1220	0.1347 ± 0.0102
2.5	GNN trained	0.8060	0.8070	0.8140	0.8090 ± 0.0036
2.6	GAT trained	0.8080	0.8050	0.8020	0.8050 ± 0.0024

Table 1: The accuracy of the models

The GNN Encoder employs two GCN layers. It takes node features as input, processes them through a hidden layer of dimension 128, and produces an output embedding of dimension 64. We utilize the T-SNE dimensional reduction technique to visualize these embeddings, alongside visualizations of both random and original vector representations for comparison.

Subsequently, we constructed an MLP embedding to match the hidden and output dimensions of the GNN and trained a Random Forest classifier on these embeddings, as well as on random feature

Preprint. Under review.

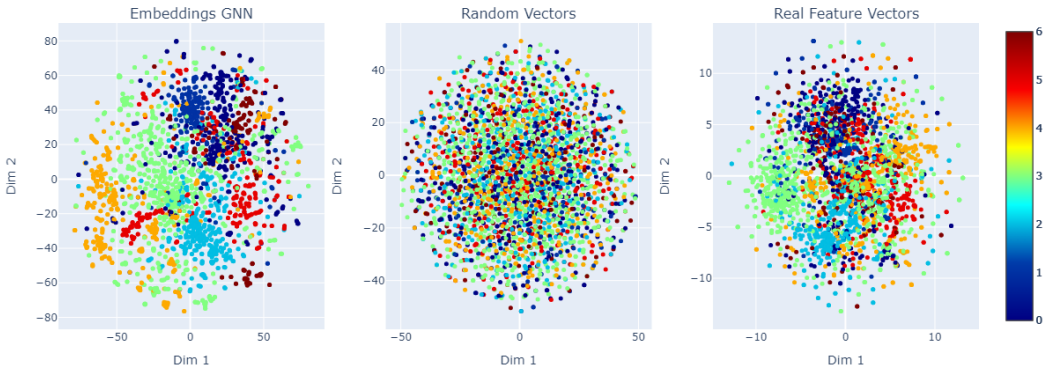


Figure 1: T-SNE dimensionality reduction

vectors of equivalent size. While the prediction accuracy on the GNN with 0.5293 ± 0.0066 remains modest, it is notably higher than that of the MLP with 0.3053 ± 0.0049. The MLP omits edge information, underscoring the significance of edge information in the prediction task.

Trained GNN: The trained GNN of the same network size and tuned hyper parameter, achieves an average classification accuracy of 0.8090 ± 0.0036. To gain deeper insights into the information flow, we introduce Graph Attention Network (GAT) layers into our previously established GNN architecture. We trained the model, achieving an accuracy surpassing 0.8050 ± 0.0024 on the test set.

Post-training, attention weights were extracted. The attention weights are averaged over the number of attention heads over the three runs. We represent the weights of the first and second layer. In our visualization, the attention weights for a node's immediate neighbors are denoted by line thickness. We also display the ground-truth labels of neighboring nodes and the selected test node. While the first visualization shows the information from its 1-hop neighbors, the second visualization extends to information aggregation of 2-hop neighbors, revealing an increase in attention weights.

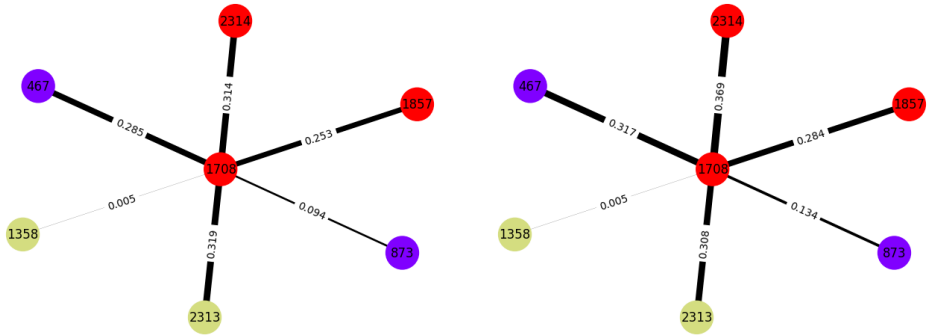


Figure 2: Attention weight visualization of node 1708

2 Task 3: Inductive Learning

Before we train any model, we try to understand the dataset at hand and a few key characteristics. The *ENZYMES* dataset contains 600 graphs. For each of the 6 class labels, we have 100 samples, so the dataset is perfectly balanced. Each node represents secondary structure elements of the encoded macromolecule, i.e. the node is a helix, sheet, or turn. An edge connects two nodes if they are neighbors along the amino acid sequence or one of three nearest neighbors in space [Morris et al., 2020]. As the node features are only of dimensionality $d = 3$, learning the topology of the graph is more important than learning a powerful aggregation over the node features for the graph. Xu et al. [2019] discuss the expressiveness of different GNN architectures by comparing their capability to identify graphs that are topologically identical. Taking inspiration from the Weisfeiler-Lehman test,

they derive the Graph Isomophism Network (GIN). We can observe empirically that such a network is well suited for the task at hand by comparing the achieved accuracy between a simple GCN and the GIN.

Model	Accuracy			
	Seed			$\mu \pm \sigma$
	42	43	44	
GCN	0.312	0.268	0.228	0.269 ± 0.034
GIN [Xu et al., 2019]	0.579	0.449	0.519	0.516 ± 0.053
HPGSL [Zhang et al., 2019]	0.274	0.281	0.303	0.286 ± 0.012
SP-MPNN [Abboud et al., 2022]	0.310	0.258	0.171	0.246 ± 0.057

Table 2: The accuracy of the models

Data Augmentation is often used when working on small datasets and with Deep Learning models that tend to work better the more data is available. In a recent paper, Heo et al. [2023] propose an augmentation strategy for graph datasets and compare accuracy for different strategies on the ENZYMES dataset. None of the reported augmentation strategies are able to improve accuracy by more than 1 to 2%, which is why we try to focus on different architectures.

We observe that due to the small dataset size and its heterogeneity, the results are very sensitive to the generated test-/train-splits (in some cases up to 15% in accuracy for the same model and hyperparameters). If we would not be constrained to use different random splits for the evaluation, it would be advisable to purposefully design a well-performing test-/train-split to minimize generalization error and potentially apply variants of curriculum learning for the best performance. We tried to implement architectures that reported improvements over the GIN architecture. Unfortunately, we were not successful in reproducing the results reported by the papers.

3 Task 4: Custom Message Passing

The results for the different experiments are provided in Table 3.

Model	Accuracy			
	Seed			$\mu \pm \sigma$
	42	43	44	
Custom Message Passing	0.3073	0.3368	0.3498	0.3313 ± 0.018
SAGEConv	0.3351	0.3151	0.3637	0.3380 ± 0.020
GRU	0.2170	0.2031	0.1554	0.1918 ± 0.026
Weighted Aggregations	0.5165	0.3151	0.3151	0.3822 ± 0.095

Table 3: The accuracy of the models

Our custom message passing layer demonstrates a performance profile that closely mirrors that of the SAGEConv layer, which aligns with our initial expectations. When we employ a non-permutation invariant aggregator, such as the GRU, within the message passing process, we observe a substantial and detrimental drop in performance. This highlights the critical importance of maintaining permutation invariance in the aggregator to achieve optimal results in the network.

As a lightweight version of the proposal of [Corso et al., 2020] to use multiple aggregation functions, we compute the mean, max, min and sum as different aggregator and propose to learn a simple weighted combination of these aggregations as the final aggregation. This way, the network can learn which aggregators to consider to which degree for the given dataset. We add the results for this classifier to the table as *Weighted Aggregations*.

A Analysis of Cora dataset

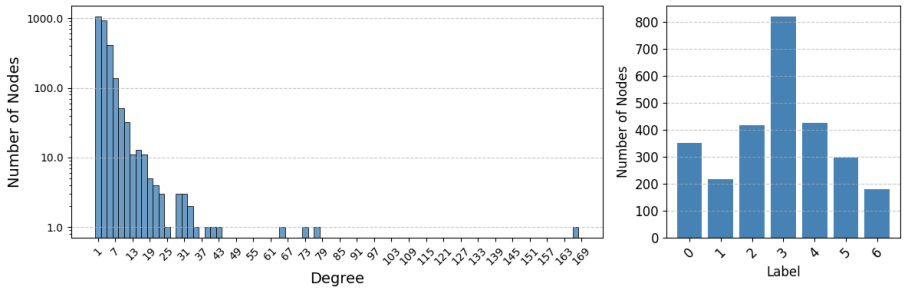
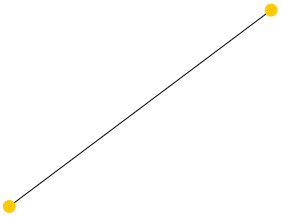
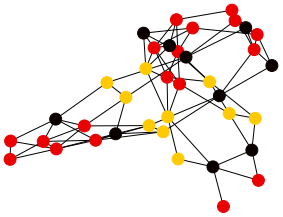


Figure 3: Left: Node Degree Distribution, Right: Distribution of Node Labels

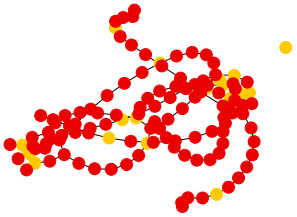
B Analysis of ENZYMES dataset



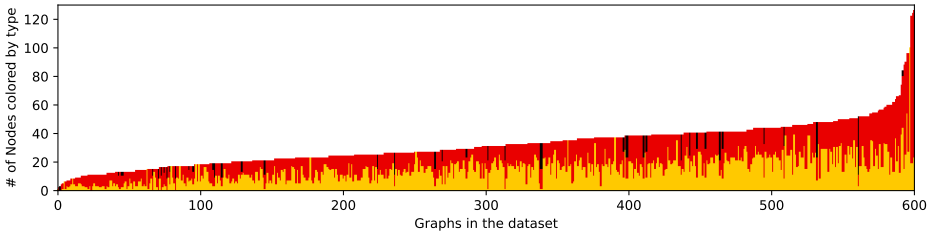
(a) Smallest graph in the dataset at index 0.



(b) Mid-sized graph with all three class labels at index 397.



(c) The largest graph in the dataset at position 599.



(d) An overview of the graphs in the dataset. The length of the line indicates the number of nodes while the colors encode the share of node types per graph.

Figure 4: Three sample graphs from the ENZYMES dataset, together with a aggregated view on the data. The colors indicate the node labels.

References

Ralph Abboud, Radoslav Dimitrov, and Ismail Ilkan Ceylan. Shortest path networks for graph property prediction. In *Learning on Graphs Conference*, pages 5–1. PMLR, 2022.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

Jaeseung Heo, Seungbeom Lee, Sungsoo Ahn, and Dongwoo Kim. Epic: Graph augmentation with edit path interpolation via learnable cost. *arXiv preprint arXiv:2306.01310*, 2023.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.