

A Comparison to Other Datasets

We aim to maintain the same graph structure as previous versions of our three datasets described below. This is to keep the number of varying factors to a minimum and just focus on how the embeddings effect the result. To achieve this we use the same adjacency matrices published with these prior datasets. Although we were unable to source all the raw data these deviations are limited.

Due to the importance of f_e on GNN performance there is a lot to discuss about prior datasets that exist within the space of GNNs in regards to these functions.

A.1 Pytorch Geometric

Pytorch Geometric python library provides a standard interface on top of Pytorch to allow for the development of graph based machine learning. The library also provides a sample of datasets from previous papers published in this field.

As is clear from the table the current standard embedding for datasets is bag of words. In the cases where bag of words approaches are not used the approach is grounded in classical text representations such as n-grams and word vectors.

The tasks in these popular datasets are node classification where the node data is frequently text. We therefore say that on the node level these are text classification tasks. The only instance of a non-text classification task is Flickr [12], though based on the fact that the underlying data is image descriptions this could also be considered a text classification task.

This demonstrates how limited the reach of GNNs currently stand as they are being trained on datasets that behave very similarly where the only difference is the specifics of the available data. We feel that this does not therefore fully test the capabilities of GNNs and puts too much emphasis on bag of words and text classification.

A.2 Open Graph Benchmark

The results in this paper focus on *node property prediction* as the data that unconnected models ordinarily work on is easily transferred to nodes in a graph. So when discussing Open Graph Benchmark [4] the focus is on the node property prediction subset (OGBN).

The goal of OGB is to create a standard set of datasets that can be used to compare different GNN architectures so a discussion as to way we did not use their datasets is warranted. The available datasets *ogbn-products*, *ogbn-proteins*, *ogbn-arxiv*, *ogbn-papers100M* and *ogbn-mag* all use variations on the same text representations used in Appendix A.1. These include Bag of Words (BoW), word2vec and skip-gram. This means the same discussions on these classical text holds here.

We see that the majority of the tasks focus on text classification, excluding *ogbn-protein*, this again draws into question how well these datasets are testing the range of classification tasks. Further to this, focusing mainly on BoW style embeddings raises the question of whether we are building good BoW extractors or graph information extractors.

A.3 Flickr

The prior Flickr dataset used in Zeng *et al.* [12] originated from McAuley *et al.* [7] which aimed to utilize network connections and image descriptions rather than the images themselves. The specific embedding function that the paper used is Bag of Words.

This embedding function is a valid representation of images but it is not easily applicable to other image datasets. Thus GNNs trained on this dataset are confined to images with descriptions that have been transformed using the same top 500 words. Noting that this list of top 500 words is not readily available.

A.4 Amazon

Zeng *et al.* [12] also provide an Amazon dataset (AmazonProducts) covering the entirety of Amazon. Without a known source we instead use available Amazon databases online to download and generate our own dataset. The embedding function used is to tokenise the reviews by 4-grams and take the

single value decomposition. This is, as with Flickr, not easily applicable outside of the original dataset.

An alternative Amazon dataset (Amazon) is also available from *Shchur et al.* [9] created originally in *McAuley et al.* [8]. Though the original source of the dataset used a pre-trained Caffe model to embed the product images this dataset did not use these. Instead they created their own embeddings using the bag of words standard with the product reviews as the raw data.

B Further Results

Table 3 contains the further results collected for AmazonInstruments

Table 3: Test accuracy on AmazonInstruments with different embeddings demonstrating how the different embeddings effects the performance and relative ranking of GNN models. Included is the standard deviation of each result.

Model	Embedding Styles			
	Bag of Words	Byte Pair	roBERTa Encoded	roBERTa
GCN	64.0% \pm 0.5	20.8% \pm 0.3	20.4% \pm 0.8	20.4% \pm 0.8
GAT	79.3% \pm 0.6	21.6% \pm 0.9	47.5% \pm 1.9	46.1% \pm 4.3
GAT2	79.4% \pm 0.3	21.2% \pm 0.6	49.8% \pm 5.0	47.8% \pm 2.8
GraphSAGE (Random)	67.5% \pm 0.3	23.9% \pm 0.6	45.1% \pm 1.2	41.9% \pm 0.6
GraphSAGE (Neighbour)	72.6% \pm 0.3	43.4% \pm 0.5	62.4% \pm 0.5	59.9% \pm 0.6

C Hyperparameters

Table 4 details the layers of each model used providing the output hidden features of each layer, the sampler used (the specifics shown in Table 5) and the maximum and minimum learning rates. Where there is a difference in learning rates we use a learning rate scheduler that decreases the learning rate when validation accuracy plateaus. Where two models use the same sampler the parameters of those samplers are identical to keep consistency across the tests.

For GraphSAINTSampler all setups use a walk length of 2 with 5 steps sampling 100 nodes per node for normalisation calculation.

Table 4: Model architecture, sampler and learning rate

Model	Hidden Features	Sampler	Learning Rate	
			Max.	Min.
GCN	256 256	Random Node	1e-2	1e-2
GAT	256 256	GraphSAINT RW	1e-2	1e-2
GAT2	256 256	GraphSAINT RW	1e-2	1e-2
GraphSAGE (Random)	256 256	Random Node	1e-3	1e-3
GraphSAGE (Neighbour)	256 256	Neighbour	1e-3	1e-3
ResNet18	<i>as provided</i>	-	1e-4	5e-6
ResNet50	<i>as provided</i>	-	1e-4	5e-6
VGG16	<i>as provided</i>	-	1e-4	5e-6

Table 5: Sampler parameters

Sampler	Dataset Split	Setup
GraphSAINT RW [12]	Train	roots: 6000
	Validation	roots: 1250
	Test	roots: 2000
Random Node	Train	# partitions:512
	Validation	# partitions:128
	Test	# partitions:256
Neighbour [2]	Train	# neighbours:[25, 10], batch size:512
	Validation	# neighbours:[25, 10], batch size:128
	Test	# neighbours:[25, 10], batch size:256