Evaluating Hybrid Active Learning Strategies Across GNN Architectures for Graph Classification

Smayan A Khanna
Department of Computer Science
University of Chicago
Chicago, IL
`smayan@uchicago.edu`

Abstract

Graph Neural Networks (GNNs) have emerged as powerful models for representation learning on relational data, achieving strong performance on tasks such as graph classification. However, the growing size of graph datasets and the scarcity of labeled data pose challenges to scalability and generalization. In domains where labeling graph instances is expensive—such as scientific applications—deciding which graphs to label becomes a key question. In this work, we investigate whether Deep Active Learning (DAL) can improve label efficiency for GNNs in graph classification. We conduct a systematic study of several acquisition strategies across multiple GNN architectures and benchmark scientific graph datasets, simulating data-constrained regimes encountered in practice.

1 Introduction

Graph-structured data is common in scientific fields such as chemistry, bio-informatics, and biophysics. Molecules, for instance, can be naturally represented as graphs, where nodes correspond to atoms and edges to chemical bonds. Similarly, cellular and tissue structures often exhibit complex, non-Euclidean topologies that are well-captured by graphs [1, 2]. In recent years, Graph Neural Networks (GNNs) have emerged as powerful models for learning on such relational data, achieving state-of-the-art performance in tasks like molecular property prediction, materials design, and disease classification [1, 2, 3].

This growing interest in GNNs for scientific applications highlights the need for sample-efficient learning strategies, particularly in domains where labeled data is scarce or expensive to obtain. Deep Active Learning (DAL) offers one such approach by iteratively selecting the most informative samples for annotation, allowing models to achieve high performance while labeling only a small, strategically chosen subset of the data [4, 5]. Numerous papers have demonstrated that DAL methods can improve label efficiency in scientific GNN applications such as molecular property prediction and materials discovery by leveraging uncertainty- or diversity-based sampling to reduce annotation cost without sacrificing performance [6, 7, 8] .

While much of the active learning (AL) literature on GNNs has focused on node classification [4, 9, 10, 11], graph-level classification — where each data point is an entire graph — remains comparatively underexplored. Although recent work has proposed novel AL and self-supervised strategies tailored to specific graph classification tasks, these methods often emphasize one particular approach or architecture [7, 12]. In contrast, our goal is to systematically evaluate traditional deep active learning (DAL) strategies across multiple GNN architectures and datasets, focusing on their effectiveness in graph-level scientific domains.

Preprint. Under review.

To this end, we apply standard DAL acquisition methods across several widely used bio-molecular datasets from the TU-Dortmund collection [13]. Our findings offer a comparative analysis of how DAL performs under varying model and data regimes, aiming to provide a practical baseline and proof-of-concept for future work on DAL-aided scientific discovery.

2 Related Work

This paper draws mainly on two areas of literature: Deep Active Learning and Graph Neural Networks.

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) are deep learning models designed to operate on graph-structured data. Given a graph $G = (\mathbf{X}, \mathbf{A})$, where $\mathbf{X} \in \mathbb{R}^{N \times D}$ is a node feature matrix and $\mathbf{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix, GNNs iteratively compute node representations by aggregating information from local neighborhoods [14]. This message-passing process is conventionally written as:

$$\mathbf{m}_u^{(l)} = \text{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), \quad u \in \mathcal{N}(i) \cup \{i\} \tag{1}$$

$$\mathbf{h}_i^{(l)} = \text{AGG}^{(l)}\left(\mathbf{h}_i^{(l-1)}, \left\{\mathbf{m}_u^{(l)} : u \in \mathcal{N}(i)\right\}\right) \tag{2}$$

where $\mathbf{m}_u^{(l)}$ denotes the message computed by node u at layer l, and $\mathbf{h}_i^{(l)}$ is the updated representation of node i at the same layer. The functions $\text{MSG}^{(l)}$ and $\text{AGG}^{(l)}$ denote the message and aggregation functions at layer l, respectively.

In this work, we focus on graph-level classification, where node embeddings are pooled to produce a graph representation. We evaluate several common GNN architectures that differ in how they aggregate neighborhood information:

- **GCN** [15]: Performs a normalized average over neighboring features using the graph Laplacian.
- **GAT** [16]: Uses self-attention to weight neighbor features adaptively.
- **GraphSAGE** [17]: Aggregates sampled neighbors via functions like mean or max and supports inductive inference.

These models vary in expressivity and inductive bias, which may impact how effective active learning is across architectures. By comparing acquisition strategies across these backbones, we aim to assess whether DAL effectiveness is architecture-sensitive.

2.2 Deep Active Learning

DAL aims to reduce labeling costs by iteratively selecting the most informative samples from a large pool of unlabeled data [4, 9, 18]. Previous works on DAL for graph classification have explored using contrastive objectives or self-supervised pretraining to improve sample efficiency, often designing task-specific acquisition functions that balance uncertainty and diversity. For instance, [12] use a semi-supervised active learning framework that combines graph contrastive learning with both local and global selection strategies to improve graph classification . In, [7], a principled active learning method is introduced specifically for 3D molecular graphs, integrating geometric diversity and uncertainty estimation. However, both works evaluate an acquisition strategy on a fixed GNN architecture. In contrast, our work systematically compares multiple DAL acquisition functions across several standard GNN models and benchmark datasets. We hope to assess whether standard DAL techniques generalize well across architectures and data regimes relevant to scientific problems.

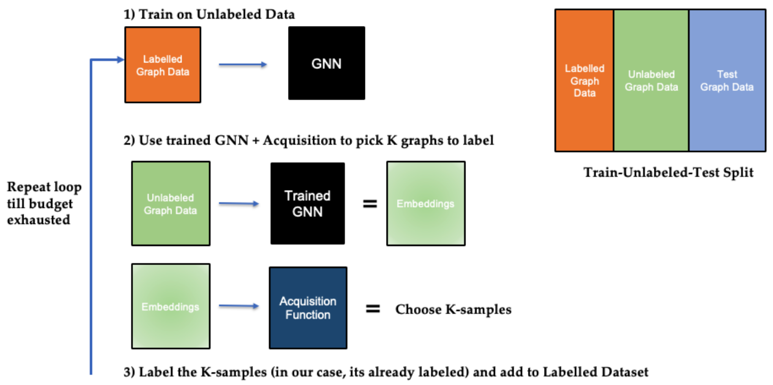
1) Train on Unlabeled Data
Labelled Graph Data
GNN
2) Use trained GNN + Acquisition to pick K graphs to label
Repeat loop till budget exhausted
Unlabeled Graph Data
Trained GNN
Embeddings
Embeddings
Acquisition Function
Choose K-samples
3) Label the K-samples (in our case, its already labeled) and add to Labelled Dataset
Labelled Graph Data
Unlabeled Graph Data
Test Graph Data
Train-Unlabeled-Test Split

Figure 1: Overview of our active learning pipeline for graph classification. Step 1: Given an initial defined train-unlabeled and test split of our dataset, we begin with a small labeled subset and iteratively train a GNN model on it. Step 2: The trained GNN generates embeddings for the unlabeled graphs, which are passed to an acquisition function to select the next batch of K most informative samples. Step 3: These are then added to the labeled set, and the process repeats until the labeling budget is exhausted. A fixed test set is held out throughout for evaluation.

3 Proposed Method

Our goal is to evaluate how different Deep Active Learning (DAL) acquisition strategies affect performance on graph classification tasks using different GNN architectures. We adopt a pool-based DAL framework, where each data point is a graph $G_i = (X_i, A_i)$ with an associated label $y_i \in \{1, \dots, C\}$. At each round, a GNN is trained on a small labeled subset and used to query new graphs from the unlabeled pool via an acquisition function $\alpha(\cdot)$. The newly labeled samples are added to the training set, and the model is fine-tuned. This process repeats until a fixed labeling budget is exhausted. An overview of this setup is shown in Fig. 1.

3.1 Acquisition Functions

We investigate the following acquisition functions for our DAL loop:

Random Sampling Serves as a standard baseline. At each iteration, K unlabeled graphs are selected uniformly at random.

Entropy Sampling [19] Uses the model's predictive uncertainty. For each graph G_i, we compute the predictive class probabilities $\hat{p}_i \in \mathbb{R}^C$ using the softmax of the output logits. The entropy is given by:

$$H(G_i) = -\sum_{c=1}^{C} \hat{p}_{i,c} \log \hat{p}_{i,c} \tag{3}$$

K-graphs with the highest entropy are selected for labeling. This is known in DAL literature as an uncertainty-based sampling method, as the entropy quantifies the model's confidence in its predictions.

Embedding Clustering (KMeans) [20] We compute graph-level embeddings of the unlabeled dataset using our GNN and apply KMeans clustering to the embeddings. From each cluster, we sample points closest to the centroid.

Embedding Similarity Sampling After training the GNN at the t^{th} active learning round, we compute graph-level embeddings $r_i \in \mathbb{R}^d$ for all graphs in the labeled set and average them to obtain a representative embedding:

$$\bar{\mathbf{r}}_t = \frac{1}{N} \sum_{i=1}^{N} r_i \tag{4}$$

Next, for each graph G_j in the unlabeled pool, we compute the cosine similarity between its embedding r_j and the labeled average embedding $\bar{\mathbf{r}}_t$:

$$\text{sim}(r_j, \bar{r}_t) = \frac{r_j \cdot \bar{r}_t}{\|r_j\| \, \|\bar{r}_t\|} \tag{5}$$

The top-K graphs with the highest cosine similarity scores are selected for labeling. This method prioritizes similarity in embedding space.

3.2 Implementation Details

We implement all GNN architectures and acquisition functions using PyTorch Geometric. During each acquisition round, the model is trained for a fixed number of epochs using the Adam optimizer and a cross-entropy loss.

For evaluation, we hold out a fixed test set comprising 30% of the dataset from the start. The initial labeled set size that we choose varies slightly across datasets (based on data constraints and other choices).

To ensure reproducibility and fairness, we fix all sources of randomness — including dataset splits, dataloader batching, and model initialization — using the same set of random seeds across all acquisition strategies. This allows us to isolate the effect of the acquisition function itself and reduce variance in performance metrics. We report the mean ± standard deviation of final test accuracy over these runs. The exact number of seeds used along with other experimental configurations are reported in the next section.

4 Experiments

We evaluate our deep active learning framework on four benchmark graph classification datasets from the TU Dortmund collection: **PROTEINS**, **DD**, **NCI1**, and **ENZYMES** [13]. These datasets represent a range of bio-physical systems to test different acquisition strategies in scientific graph learning.

4.1 Datasets

- **PROTEINS**: A dataset of protein graphs where nodes represent secondary structure elements (SSEs) and edges represent neighborhood relations in 3D space. The task is to classify proteins into enzyme and non-enzyme classes. It contains 1113 graphs with a moderate average graph size of 39 nodes.
- **DD**: Derived from the D&D protein database, this dataset contains protein structures represented as graphs, with nodes as amino acids and edges indicating spatial proximity. Like PROTEINS, the task is to classify the graphs into binary enzyme and non-enzyme classes. It contains 1178 relatively large graphs (average of 284 nodes), making it a good test of scalability for DAL methods.
- **NCI1**: A molecular graph dataset of over 4000 chemical compounds screened for activity against non-small cell lung cancer and leukemia. Each graph represents a molecule, with atoms as nodes and chemical bonds as edges. NCI1 consists of binary labels and moderately sized graphs, averaging around 30 nodes. As the largest dataset in our experiments, with over 4000 graphs, it provides insight into how DAL methods scale with dataset size.
- **ENZYMES**: A multi-class classification dataset of enzyme protein tertiary structures. Each graph corresponds to a protein, and the task is to classify it into one of 6 enzyme classes. ENZYMES is a small dataset with only 600 graphs but contains high variability in structure, making it challenging for GNNs.

4.2 Experimental Setup

We follow the DAL procedure outlined in Section 1. For each dataset, we hold out 30% of the graphs as a fixed test set. The remaining 70% is used for training and active learning. The exact experimental configuration for each dataset, including the split of the labeled-unlabeled data, is shown in Table 1. Our design choices for each dataset depend on both their size and task complexity. For **PROTEINS**, we use a lightweight GNN (1 message passing layer, 32 hidden dimensions) to avoid overfitting and ensure that performance does not saturate too early, as this dataset is relatively small and binary-labeled. Since **NCI1** is the largest dataset by far, we select more samples per acquisition round (20 per round), while keeping the initial labeled set to 5%. **ENZYMES** is both small and highly variable; we therefore start with a larger labeled set (15%) and allow a larger total labeling budget (40%) to give the model enough data to learn on. For **DD**, due to the large graph sizes and moderate dataset scale, we use a slightly deeper GNN (2 layers, 64 hidden units). For all datasets except NCI1, we use 10 seeds to obtain more robust representative results. Due to the large size of the NCI1 dataset, we limited our experiments to 5 seeds. However, as shown in Figure 5, the standard deviations across runs were already low, suggesting that the large dataset size contributes to more stable and consistent performance across seeds.

Table 1: Dataset-specific experimental settings used for deep active learning. Initial labeled set size, GNN architecture hyperparameters, and labeling budgets are adjusted per dataset.

Dataset	**Initial Labeled %**	**Hidden Dim**	**# Layers**	**Labeling Budget**	**# Seeds**	**Total Rounds**
PROTEINS	10%	32	1	30% (10 per round)	10	20
DD	10%	64	2	30% (10 per round)	10	20
NCI1	5%	32	2	30% (20 per round)	5	30
ENZYMES	15%	32	2	40% (5 per round)	10	40

All experiments are run with GCN, GAT, and GraphSAGE GNN architectures. We train our models using NVIDIA GPU's granted to us by UChicago's RCC (Research Computing Cluster).

4.3 Results

We summarize the main findings of our experiments in Tables 1–4, reporting the final test accuracy achieved by each acquisition strategy across the four datasets (NCI1, ENZYMES, PROTEINS, and DD) and three GNN architectures (GCN, GAT, and GraphSAGE).

Since we are testing DAL strategies on datasets which are fully labeled, we can also compare the performance of GNN models trained with active learning to those trained on the full dataset (with the total number of training gradient steps kept fixed to ensure fairness in comparison). This provides a natural upper bound for performance, allowing us to quantify how close each acquisition strategy comes to full supervision. In many cases, we find that DAL methods remarkably approach full-dataset accuracy using significantly fewer labeled examples. This serves as a proof-of-concept that DAL can reduce labeling costs (in scenarios where there is a non-zero label cost) for graph representation learning for science.

Our experiments reveal that no single acquisition strategy consistently dominates across all datasets and GNN architectures. However, several notable patterns emerge:

- **Entropy sampling** often delivers strong performance. It achieves the best accuracy on the DD dataset with both GAT and GraphSAGE (Table 3) and performs competitively on ENZYMES using GAT (Table 5).
- **Embedding similarity**, our proposed heuristic, shows mixed yet insightful behavior. It excels on ENZYMES using GraphSAGE (Table 5) but underperforms on structurally complex datasets like NCI1. This suggests the method's sensitivity to embedding stability and to the representativeness of the initial labeled pool. Since acquisition decisions are based on cosine similarity between mean labeled embeddings and unlabeled candidates, early sampling bias can significantly affect subsequent rounds.
- **KMeans clustering** performs especially well on PROTEINS with GCN and GAT (Table 2), suggesting that geometric structure in embedding space may be informative for early rounds of acquisition in smaller, binary-labeled datasets.
- Interestingly, **random sampling** remains surprisingly competitive—particularly on the large-scale NCI1 dataset (Table 4). This raises important questions about the marginal benefit of DAL strategies in high-volume, structurally diverse domains.

Table 2: Test accuracy at final acquisition round on the PROTEINS dataset. Values are mean ± std over seeds; best strategy per model bolded. The full-dataset result is provided for reference.

Results on PROTEINS Dataset: Test Accuracy			
Strategy	**GCN**	**GAT**	**SAGE**
Entropy	0.6715 ± 0.0238	**0.6622 ± 0.0256**	**0.6718 ± 0.0229**
Kmeans	**0.6751 ± 0.0260**	0.6577 ± 0.0204	0.6610 ± 0.0215
Random	0.6583 ± 0.0317	0.6598 ± 0.0240	0.6556 ± 0.0229
Embedding	0.6411 ± 0.0297	0.6495 ± 0.0261	0.6378 ± 0.0251
Full Dataset	0.6799 ± 0.0250	0.6742 ± 0.0277	0.6832 ± 0.0192

Table 3: Test accuracy at final acquisition round on the DD dataset. Values are mean ± std over seeds; best strategy per model bolded. The full-dataset result is provided for reference.

Results on DD Dataset: Test Accuracy			
Strategy	**GCN**	**GAT**	**SAGE**
Entropy	0.6980 ± 0.0209	0.6915 ± 0.0200	**0.6969 ± 0.0195**
Kmeans	0.6875 ± 0.0246	0.6816 ± 0.0170	0.6941 ± 0.0277
Random	0.6909 ± 0.0272	0.6841 ± 0.0252	0.6926 ± 0.0175
Embedding	**0.7000 ± 0.0273**	**0.6926 ± 0.0268**	0.6909 ± 0.0260
Full Dataset	0.7139 ± 0.0189	0.7116 ± 0.0120	0.7227 ± 0.0094

Table 4: Test accuracy at final acquisition round on the NCI1 dataset. Values are mean ± std over seeds; best strategy per model bolded. The full-dataset result is provided for reference.

Results on NCI1 Dataset: Test Accuracy			
Strategy	**GCN**	**GAT**	**SAGE**
Entropy	0.6887 ± 0.0050	0.6920 ± 0.0054	0.7118 ± 0.0036
Kmeans	0.6697 ± 0.0047	0.6722 ± 0.0121	0.7011 ± 0.0088
Random	**0.6952 ± 0.0120**	**0.6968 ± 0.0113**	**0.7205 ± 0.0158**
Embedding	0.6827 ± 0.0110	0.6631 ± 0.0187	0.6842 ± 0.0204
Full Dataset	0.7045 ± 0.0119	0.7038 ± 0.0067	0.7213 ± 0.0057

Table 5: Test accuracy at final acquisition round on the ENZYMES dataset. Values are mean ± std over seeds; best strategy per model bolded. The full-dataset result is provided for reference.

Results on ENZYMES Dataset: Test Accuracy			
Strategy	**GCN**	**GAT**	**SAGE**
Entropy	0.2494 ± 0.0361	0.2500 ± 0.0290	**0.2839 ± 0.0249**
Kmeans	0.2339 ± 0.0277	0.2389 ± 0.0416	0.2556 ± 0.0226
Random	0.2511 ± 0.0243	0.2539 ± 0.0245	0.2728 ± 0.0201
Embedding	**0.2644 ± 0.0344**	**0.2572 ± 0.0348**	0.2817 ± 0.0435
Full Dataset	0.2706 ± 0.0194	0.2706 ± 0.0242	0.2894 ± 0.0325

4.4 Visualizing Acquisition Dynamics

We produced accuracy-over-acquisition-round plots for each dataset and model combination (a total of 12 plots). We only include representative plots for the DD dataset here in 3. The complete set is provided in the appendix section.

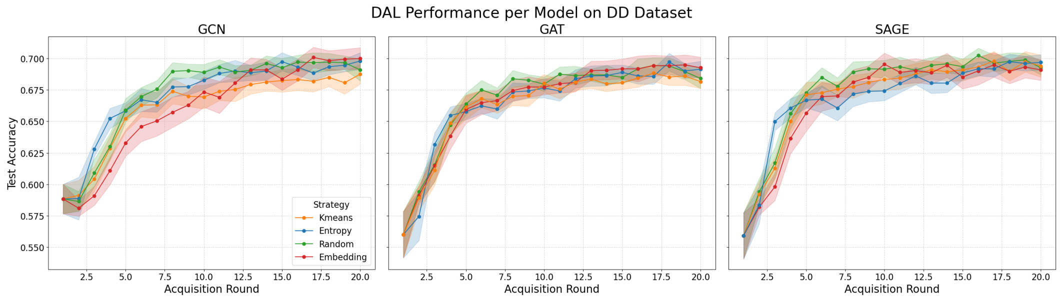

Figure 2: Test accuracy vs. acquisition rounds on the DD dataset for GCN (left), GAT (middle), and GraphSAGE (right).

5 Discussion

5.1 Summary of Key Takeaways

Our experiments show that deep active learning (DAL) can effectively reduce labeling costs when training graph neural networks (GNNs). Specifically, DAL methods frequently achieved accuracy close to models trained on fully labeled datasets, even when using significantly fewer data samples. Interestingly, no single acquisition method consistently performed best across all datasets and architectures, indicating that the choice of strategy should depend on the specific dataset and model being used. Entropy-based uncertainty sampling emerged as consistently strong, while simpler heuristics like embedding similarity and random sampling were also surprisingly effective.

5.2 Limitations and Future Work

Despite some promising results, several limitations and opportunities for improvement emerged. First, we primarily focused on uncertainty-based (entropy) and representation-based (embedding) sampling techniques, but exploring more advanced methods like Bayesian Active Learning by Disagreement (BALD) or combinations of different strategies could further enhance label efficiency. Additionally, the representational capacity of the GNNs used (GCN, GAT, GraphSAGE) might have been limited, especially for structurally challenging datasets such as ENZYMES. Using more expressive architectures like Graph Isomorphism Networks (GINs) [21] could help the models better capture structural patterns. Finally, our current experiments employed continuous fine-tuning across acquisition rounds. Investigating a cold-start approach—where models are retrained from scratch after each acquisition—could yield further insights into learning dynamics of DAL.

References

[1] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in Genetics*, Volume 12 - 2021, 2021.

[2] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.

[3] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[4] Dongyuan Li, Zhen Wang, Yankai Chen, Renhe Jiang, Weiping Ding, and Manabu Okumura. A survey on deep active learning: Recent advances and new frontiers, 2024.

[5] Burr Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[6] Bowen Li and Srinivas Rangarajan. A diversity maximizing active learning strategy for graph neural network models of chemical properties. *Mol. Syst. Des. Eng.*, 7:1697–1706, 2022.

[7] Ronast Subedi, Lu Wei, Wenhan Gao, Shayok Chakraborty, and Yi Liu. Empowering active learning for 3d molecular graphs with geometric graph isomorphism. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 55507–55537. Curran Associates, Inc., 2024.

[8] Stephan Thaler, Felix Mayr, Siby Thomas, Alessio Gagliardi, and Julija Zavadlav. Active learning graph neural networks for partial charge prediction of metal-organic frameworks via dropout monte carlo. *npj Computational Materials*, 10(1):86, 2024.

[9] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. Active learning for graph embedding, 2017.

[10] Zixing Song, Yifei Zhang, and Irwin King. No change, no gain: Empowering graph neural networks with expected model change maximization for active learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 47511–47526. Curran Associates, Inc., 2023.

[11] Georgios Katsimpras and Georgios Paliouras. Improving graph neural networks by combining active learning with self-training. *Data Mining and Knowledge Discovery*, 38(1):110–127, 2024.

[12] Wei Ju, Zhengyang Mao, Ziyue Qiao, Yifang Qin, Siyu Yi, Zhiping Xiao, Xiao Luo, Yanjie Fu, and Ming Zhang. Focus on informative graphs! semi-supervised active learning for graph-level classification. *Pattern Recognition*, 153:110567, 2024.

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

[14] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

[15] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[16] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[17] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[18] Kuan-Hao Huang. Deepal: Deep active learning in python. *arXiv preprint arXiv:2111.15258*, 2021.

[19] Burr Settles. Active learning literature survey. 2009.

[20] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach, 2018.

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

A Technical Appendices and Supplementary Material

A.1 Details on Specific GNN

Graph Attention Networks (GATs) [16]: Use learned attention weights to perform a weighted sum over neighbor features:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)}\right) \tag{6}$$

where α_{ij} are normalized attention coefficients learned via self-attention over neighboring nodes.

GraphSAGE [17]: Samples neighbors and aggregates their features using a function AGG, which can be mean, max, or even an LSTM:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \left[\mathbf{h}_i^{(l)} \| \text{AGG}_{j\in\mathcal{N}(i)}\left(\mathbf{h}_j^{(l)}\right)\right]\right) \tag{7}$$

where $\|$ denotes concatenation and σ is an activation function.

Different GNN architectures vary in their inductive biases and expressiveness, so evaluating DAL across multiple models helps ensure that observed trends are not architecture-specific.

A.2 Graph Classification with GNNs

GNNs obtain node embeddings for each node in the graph via message-passing, as described earlier. In order to do graph classification, node embeddings are aggregated or "pooled". In our case, we use the global-mean-pool formula, which simply averages node embeddings:

$$\mathbf{r} = \frac{1}{N}\sum_{n=1}^{N} h_n \tag{8}$$

where r is the graph embedding for graph $\mathcal{G}$, N is the number of nodes in the graph, and h_n is the nth node embedding. To obtain a graph classification, the embedding $\mathbf{r}$ is passed through a linear classifier that outputs a logit vector $\mathbf{z} \in \mathbb{R}^C$, where C is the number of classes. The predicted label is the index corresponding to the highest logit.

A.3 Plots for all Datasets

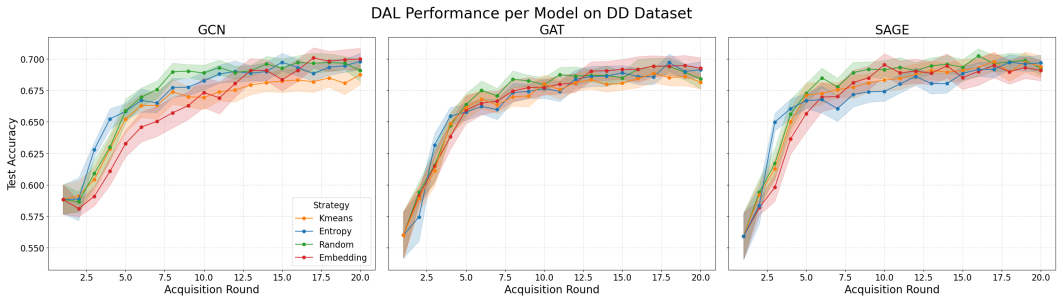

Figure 3: Test accuracy vs. acquisition rounds on the DD dataset for GCN (left), GAT (middle), and GraphSAGE (right).

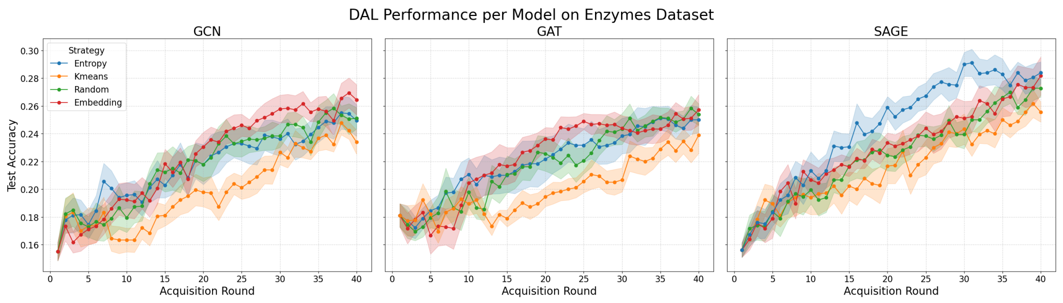

Figure 4: Test accuracy vs. acquisition rounds on the enzymes dataset for GCN (left), GAT (middle), and GraphSAGE (right).

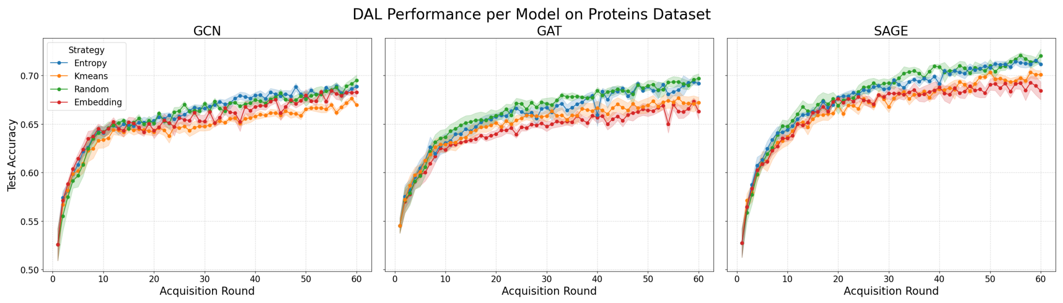

Figure 5: Test accuracy vs. acquisition rounds on the NCI1 dataset for GCN (left), GAT (middle), and GraphSAGE (right).

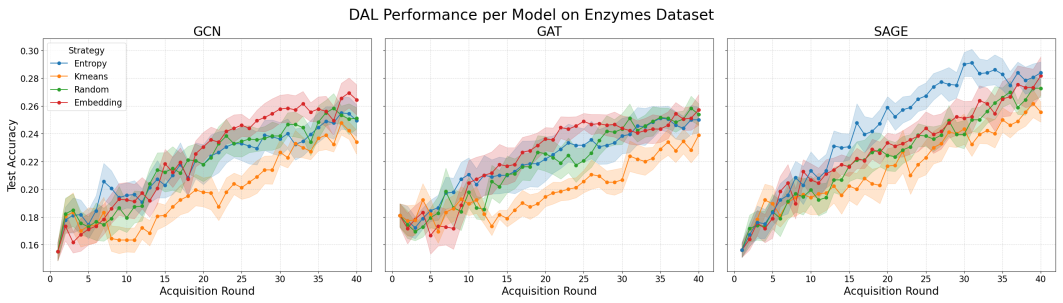

Figure 6: Test accuracy vs. acquisition rounds on the Proteins dataset for GCN (left), GAT (middle), and GraphSAGE (right).