DiffusAL: Coupling Active Learning with Graph Diffusion for Label-Efficient Node Classification

Sandra Gilhuber[1,2(✉)], Julian Busch[1,3], Daniel Rotthues[1], Christian M. M. Frey[4], and Thomas Seidl[1,2,4]

[1] LMU Munich, Munich, Germany
{gilhuber,seidl}@dbs.ifi.lmu.de
[2] Munich Center for Machine Learning (MCML), Munich, Germany
[3] Siemens Technology, Princeton, NJ, USA
busch.julian@siemens.com
[4] Fraunhofer IIS, Erlangen, Germany
christian.maximilian.michael.frey@iis.fraunhofer.de

Abstract. Node classification is one of the core tasks on attributed graphs, but successful graph learning solutions require sufficiently labeled data. To keep annotation costs low, active graph learning focuses on selecting the most qualitative subset of nodes that maximizes label efficiency. However, deciding which heuristic is best suited for an unlabeled graph to increase label efficiency is a persistent challenge. Existing solutions either neglect aligning the learned model and the sampling method or focus only on limited selection aspects. They are thus sometimes worse or only equally good as random sampling. In this work, we introduce a novel active graph learning approach called *DiffusAL*, showing significant robustness in diverse settings. Toward better transferability between different graph structures, we combine three independent scoring functions to identify the most informative node samples for labeling in a parameter-free way: i) *Model Uncertainty*, ii) *Diversity Component*, and iii) *Node Importance* computed via graph diffusion heuristics. Most of our calculations for acquisition and training can be pre-processed, making DiffusAL more efficient compared to approaches combining diverse selection criteria and similarly fast as simpler heuristics. Our experiments on various benchmark datasets show that, unlike previous methods, our approach significantly outperforms random selection in 100% of all datasets and labeling budgets tested.

Keywords: active learning · node classification · graph neural networks

1 Introduction

Graph representation learning [17] and, especially, *Graph Neural Networks* (GNNs) [2,5,16] have been adopted as a primary approach for solving machine

S. Gilhuber and J. Busch—Equal contribution

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14169, pp. 75–91, 2023.
https://doi.org/10.1007/978-3-031-43412-9_5

learning tasks on graph-structured data, including node classification [18], graph classification [21], and link prediction [41]. Applications range from quantum chemistry [16] over traffic forecasting [44] to cyber-security [6].

However, supervised GNN models require sufficient training labels and usually assume that such labels are freely available. But, in reality, while unlabeled data is usually abundant, it is laborious and costly to provide annotations. Graph active learning has emerged as a promising direction to reduce labeling costs by carefully deciding which data should be labeled to increase label efficiency. Under a limited budget, e.g., a fixed number of data samples to be labeled or time spent labeling by a domain expert, active learning aims to annotate an optimized set of training data iteratively. Hence, a key aspect of graph active learning is identifying the most informative instances in the abundance of unlabeled data for labeling. In particular, the goal is to be consistently more label-efficient than random labeling. Since random sampling is arguably the fastest and least complex method, active learning methods that are not significantly better than random sampling are not worthwhile.

However, since graphs can vary widely, it is very difficult to design an approach significantly better than random sampling across different labeling budgets and graph structures. Existing graph-active learning approaches reach their limits for various reasons: Some approaches focus only on limited selection aspects [23,28] and outperform random selection only on certain graphs. Others focus on one-shot selection without iterative re-training and active selection and can therefore not exploit model-related uncertainty scores [37,43]. Other methods try to include various criteria in the selection but are sensitive to user-defined hyper-parameters or are not deliberately aligned with the used model architecture [8,15]. Moreover, many methods use a GCN [18] for training and acquisition. However, GCNs learn latent node features and perform neighborhood aggregation in a coupled fashion, which can negatively influence the time needed for the active learning procedure. In contrast, *Graph diffusion* is a promising direction tackling limitations such as restriction to k-hop neighborhoods [7] or over-smoothing, where neighborhood aggregation and learning are decoupled.

In this work, we use diffusion-based heuristics to combine graph learning with active learning. In particular, we propose *DiffusAL*, a novel graph active learning method that leverages graph diffusion for highly accurate node classification and efficiently re-uses the computed diffusion matrix and diffused node feature vectors in the learning procedure.

We introduce a new scoring function for identifying a node's utility which consists of three factors: i) *Model Uncertainty*, ii) *Diversity Component*, and iii) *Node Importance*. DiffusAL combines these scores in a parameter-free scoring function that naturally adapts to consecutively learning iterations.

Specifically, for i) *Model Uncertainty*, we exploit a state-of-the-art scoring that has shown an improving impact on the selection of nodes [32]. Next, the ii) *Diversity Component* refers to the variability of node features and, therefore, their respective labels. For that, we apply a clustering method on the precomputed diffusion matrix where diversity is reached by picking samples from underrepresented communities. Finally, for computing iii) *Node Importance*, we exploit the information given by diffusion matrix based on the Personalized

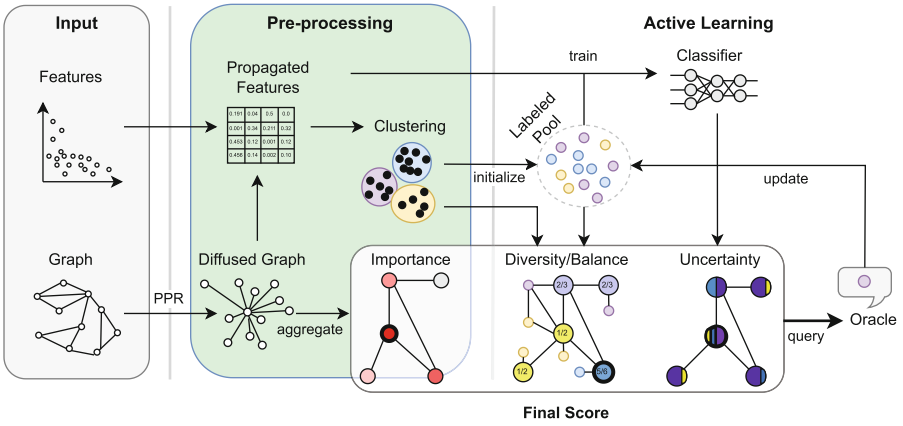


Fig. 1. DiffusAL pipeline consisting of the original input graph and corresponding node features (grey box), pre-computed static model-independent scores, such as the propagated feature matrix and derived node importance (green box), a dynamic, model-independent score based on the composition of the labeled pool (Diversity/Balance), as well as a dynamic, model-dependent informativeness score (Uncertainty). These scores are combined into a final node rating (white box) to select the most useful instances for annotation. (Color figure online)

PageRank (PPR), which provides information about the relative importance of nodes in a graph w.r.t. a particular seed node. The high-level key concepts of DiffusAL are illustrated in Fig. 1.

We evaluate DiffusAL on five real-world benchmark datasets, demonstrating its superiority over a variety of competitors. Notably, DiffusAL is the only competitor to outperform random selection with statistical significance in 100% of the evaluated datasets and labeling budgets. In a series of ablation studies, we show that DiffusAL works consistently well on all benchmark datasets, analyze which components contribute to its performance, and investigate its efficiency.

In summary, our contributions are as follows:

- Enhancing the selection of influential nodes by using *diffusion-based node importance* and utilizing pre-computed *clustering on diffused features* to prevent oversampling a particular region.
- Combining three complementary node scoring components in a parameter-free way.
- Achieving high efficiency by propagating statically pre-computed features stored in a diffusion matrix.

2 Related Work

Early works on graph active learning [3,24] exploit the graph structure for selecting nodes for querying without graph representation learning. More recent

approaches [8,15,23,28,38] use GCNs to exploit the graph structure as well as learned features. FeatProp [38] leverages node feature propagation followed by K-Medoids clustering for the selection of instances. By defining the pairwise node distances between the corresponding propagated node features, the model selects nodes being closest to the cluster representatives yielding a diverse set over the input space. However, the diversity scoring function in our model puts more weight on underrepresented clusters yielding a more balanced view of the available data space and, therefore, is more suitable for imbalanced data. In [43], the authors proposed *GRAIN*, a model inspecting social influence maximization for data selection. Their objective is a diversified influence maximization by exploiting novel influence and diversity functions. In contrast to their work, we focus on an iterative active learning setting [10] since it directly enables exploiting the uncertainty scores entangled to a model which is known to be valuable for query selection. The most related work to our approach is presented in [8] where the authors propose *Active Graph Embedding* (AGE) using as selection heuristic a weighted sum of information entropy, information density, and graph centrality defined on direct neighborhoods. For the latter, they propose to use PageRank centrality. The time-sensitive coefficients of the weighted sum are chosen from a beta distribution using the number of training iterations as input. We overcome these limitations related the restriction on direct neighborhoods aggregations used in standard GNNs [2,5,16] by leveraging continuous relationships via graph diffusion [7,20]. In [15], *ANRMAB* is proposed. It uses a multi-armed bandit mechanism for adaptive decision-making by assigning different weights to different criteria when constructing the score to select the most informative nodes for labeling. The model *LSCALE* [23] exploits clustering-based (K-Medoids) active learning on a designed latent space leveraging two properties: low label requirements and informative distances. For the latter, the authors integrate *Deep Graph Infomax* [36] as an unsupervised model. Therefore, in contrast to our approach, the model utilizes a purely distance-based selection heuristic. The method *GEEM* [29] maximizes the expected error reduction to select informative nodes to label.

To the best of our knowledge, we are the first to leverage the power of diffusion-based heuristics for the computation of node importance, being an integral part of our scoring function, combining three complementary components to compute the nodes yielding the highest utility scores. Moreover, our novel scoring function uncouples from any parameter presets, being a critical choice without any a priori knowledge about the input data.

3 DiffusAL

3.1 Preliminaries

Notation. We consider the problem of active learning for node classification. We are given a graph $G = (V, E)$ represented by an adjacency matrix $A \in \{0,1\}^{n \times n}$ along with a node feature matrix $X \in \mathbb{R}^{n \times d}$. Each node $v \in V$ belongs to exactly one class $c_v \in \{1, \ldots, C\}$, where C is the number of classes present

in the dataset. A budget constraint B denotes the maximum number of nodes for which the active learning algorithm may request the correct labels from the oracle. The main goal is to select a subset of nodes $S \subset V$ such that $|S| = B$ and the accuracy of a classification model trained on these nodes is maximized. In a batch setting, b denotes the number of nodes selected within each acquisition round.

Recap: Feature Diffusion. In contrast to conventional GNN architectures [18,35,39] that learn latent node features and perform neighborhood aggregation in a coupled fashion, *graph diffusion* effectively decouples the two steps to address certain shortcomings of conventional GNN architectures, including the restriction to k-hop neighborhoods [7] and issues related to over-smoothing[14,22,26,40]. The general effectiveness of diffusion, when paired with conventional GNN architectures, was shown in [20]. In general, a parametric diffusion matrix can be defined as

$$P = \sum_{k=0}^{\infty} \theta_k T^k, \tag{1}$$

where T is a transition matrix and θ are weighting parameters. A popular choice is *Personalized PageRank (PPR)* [4,7,11,12,19], where $T = AD^{-1}$ is the random walk matrix, D is the diagonal degree matrix, and $\theta_k = \alpha(1-\alpha)^k$. Intuitively, P_{ij} corresponds to the probability that a random walk starting at node i will stop at node j and can be interpreted as the importance of node j for node i. The restart probability $\alpha \in [0, 1]$ controls the effective size of a node's PPR-neighborhood. An approximation of the PPR matrix can be pre-computed in time $O(n)$ using push-based algorithms [7]. This approximation requires a second hyper-parameter $\varepsilon > 0$ that thresholds small entries and, thus, has a sparsification and noise reduction effect. Once computed, the PPR matrix can replace the adjacency matrix used by conventional message-passing networks for neighborhood aggregation [7,19].

3.2 Model Architecture

For DiffusAL, we propagate the original node features such that the propagated node features don't depend on any learned transformations and can be pre-computed as well. We propose a query-by-committee (QBC) approach [33], where the propagated node features are connected to an ensemble of MLP classifiers to robustify uncertainty estimation during the sample selection process compared to a commonly used single MLP. Additionally, features are diffused over multiple scales by varying the hyper-parameter α controlling the effective neighborhood size over which features are aggregated. In particular, the model predictions are given as

$$Y = predict\left(\sum_{j\in\{1,\ldots,M\}} transform_j\left(\sum_{i\in\{1,\ldots,K\}} P^{(\alpha_i)}X\right)\right), \tag{2}$$

where K denotes the number of scales, and M denotes the number of MLPs in the classification ensemble. The pre-computed diffused features are aggregated over multiple scales using the *sum* function and fed to the hidden layer of each MLP. The learned representations are then aggregated using the *sum* function and passed to the shared prediction layer. All ensemble members share the same architecture and only differ in the random initialization of their weights and biases. The QBC can be trained very efficiently with gradient descent, and, in particular, the expensive diffusion step needs to be performed only once as a pre-processing step.

3.3 Node Ranking and Selection

In addition to facilitating highly effective and efficient prediction, the previously computed diffusion matrix $P = \sum_{i \in \{1,\dots,K\}} P^{(\alpha_i)}$ and diffused features PX are reused to calculate expressive ranking scores for active node selection.

Model Uncertainty. For measuring model uncertainty, we utilize the QBC defined above. In particular, we compute the Shannon entropy over the softmaxed output distribution to determine the **uncertainty score** for node i:

$$s_{\text{unc}}(i) = -\sum_{j \in \{1,\dots,C\}} y_{ij} \log y_{ij}. \tag{3}$$

The scores are L1-normalized over all unlabeled nodes to $[0, 1]$, so all scoring functions share the same scale and can be sensibly combined.

While this score is inspired by the classical query-by-committee [33] approach, it differs in the sense that it doesn't average the softmax outputs of the individual committee members but rather considers the softmax output of a single shared prediction layer applied to aggregated latent representations. Thereby, differing predictions become more distinct in the softmax output.

Diversity Component. For the diversity component, we perform k-Means clustering on the *diffused features* with $k = b$ and assign each node a pseudo-label based on the clustering result. Note that we pre-compute these cluster assignments such that no re-computations are necessary at query time, in contrast to other approaches (e.g., based on GCNs), where updated node features would change the clustering.

The cluster-based pseudo labels are used to ensure decent coverage of the feature space. At each iteration, each node i receives a **diversity score**

$$s_{\text{div}}(i) = 1 - \frac{|c_{train}|}{|V_{train}|}, \tag{4}$$

where $c \in C$ denotes the cluster node i was assigned to, $|c_{train}|$ denotes the number of nodes in the currently labeled training set belonging to cluster c, and $|V_{train}|$ is the number of currently labeled training nodes. In short, each node in

the unlabeled pool is weighted by the relative size of its cluster in the training set, such that nodes from currently underrepresented clusters receive a higher score. In contrast to only focusing on avoiding redundancy in the current batch [1], our diversity score can also be interpreted as a balancing score ensuring that no region is over-sampled within the labeled pool.

Some existing works on graph active learning [8,15] ignore the limitations of a randomly initialized labeled pool and ensure class balance. However, this simplification is rather unrealistic in a real-world active learning setting. To overcome this limitation, we again exploit the k-Means clustering used for the diversity score and select nodes closest to centroids for the initial pool, inspired by clustering-based sampling approaches [23,37] and existing work on initial pool selection [9].

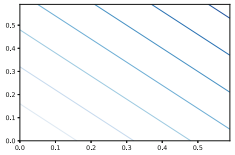
Node Importance. Graph diffusion allows for a natural way to quantify node importance. Since the weights P_{ij} used for neighborhood aggregation can be interpreted as importance scores, summing up the importance of a node i for all other nodes j yields a measure of the general importance of node i, measuring its total influence on the predictions for other nodes. Since the columns of S are stochastic, this procedure yields consistently scaled overall importance scores. In particular, the **importance score** of node i is given by the row-wise sum

$$s_{\text{imp}}(i) = \sum_{j \in V} P_{ij}. \tag{5}$$

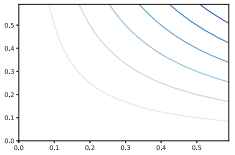
Since the importance scores for all nodes can be computed directly from the PPR matrix, they can be pre-computed before the active learning cycle starts. Our node importance score is a proxy for how much influence a node has on other nodes, where nodes with higher scores are assumed to carry more valuable information about many other nodes as well. Node importance could be interpreted as a novel representativeness measure, which has been quantified via density- or center-based selection within previous (graph) active learning approaches [8,15,42]. However, we do not need to recompute a clustering on learned representations after each selected sample, nor do we require good representations since we can extract the information directly from the graph topology. Further, our importance score of a node directly reflects the influence of that node on the model's predictions, since the weights from which we compute the scores are directly used for neighborhood aggregation. This is not the case for alternative existing measures.

Score Combination and Node Selection. In summary, the uncertainty score assigns higher weights to nodes about which the committee is most uncertain, the diversity score assigns higher weights to nodes belonging to underrepresented clusters, and the node importance score assigns higher weights to nodes with a higher influence on the predictions for other nodes. The individual scores for a node are combined in a multiplicative fashion to determine the node's utility:

$$s(i) = s_{\text{unc}}(i) \cdot s_{\text{div}}(i) \cdot s_{\text{imp}}(i). \tag{6}$$



(a) **Sum aggregation**: Isolines are straight due to fixed weighting.



(b) **Multiplicative aggregation**: Isolines are curved, favoring similar values over diverging ones.

Fig. 2. Score aggregation: for two arbitrary scores on the x and y axes (e.g. uncertainty and representativeness), the corresponding aggregated score is depicted as an isoline, i.e., each point on the line corresponds to the same final value.

As illustrated in Fig. 2, the intuition behind the multiplicative combination is to slightly favor nodes displaying a well-rounded distribution of scores over those with a strong imbalance when the sum of the scores is identical while still allowing extraordinarily important or uncertain nodes to be selected. Existing works use slightly different variations of time-sensitive weighted sums, thereby gradually shifting the focus from representativeness to uncertainty [8,42]. A disadvantage of time-sensitive weighting is that the performance of the selection algorithm depends on the choice of good hyper-parameters, which is difficult in a real-world active learning setting. In contrast, our multiplicative approach is parameter-free and naturally time-sensitive. In the early stages of training, the classifiers essentially guess predictions more or less uniformly, leading to roughly similar uncertainty scores for most nodes. Consequently, the uncertainty score is close to a constant factor applied equally to all nodes, thus naturally making the model-free scores the deciding ones in the final score. However, uncertainty scores become increasingly important once the classifiers become more confident in their predictions. The combined utility score is determined for each unlabeled node in each active learning cycle. Afterward, the unlabeled nodes are ranked according to their utility, and the nodes with the highest utility scores are labeled.

4 Experiments

To demonstrate the effectiveness and efficiency of DiffusAL, we conduct a series of experiments. In particular, we investigate three research questions:

R1 - How does DiffusAL perform compared to state-of-the-art methods?
R2 - How does each of DiffusAL's components contribute?
R3 - How is the training and acquisition efficiency?

4.1 Experimental Setup

Datasets. We evaluate DiffusAL on several well-established benchmark datasets for node classification, namely the citation networks Citeseer [30], Cora [30] and

Table 1. Dataset statistics (only considering the largest connected component).

Dataset	#Nodes	#Edges	#Features	#Classes
Citeseer	2120	3679	3703	6
Cora	2485	5069	1433	5
Pubmed	19717	44324	500	3
Co-author CS	18333	81894	6805	15
Co-author Physics	34493	247962	8415	5

Pubmed [25], as well as the co-author networks Computer Science (CS) [34] and Physics [34], summarized in Table 1. For each dataset, only the largest connected component is used, and features are L1-normalized.

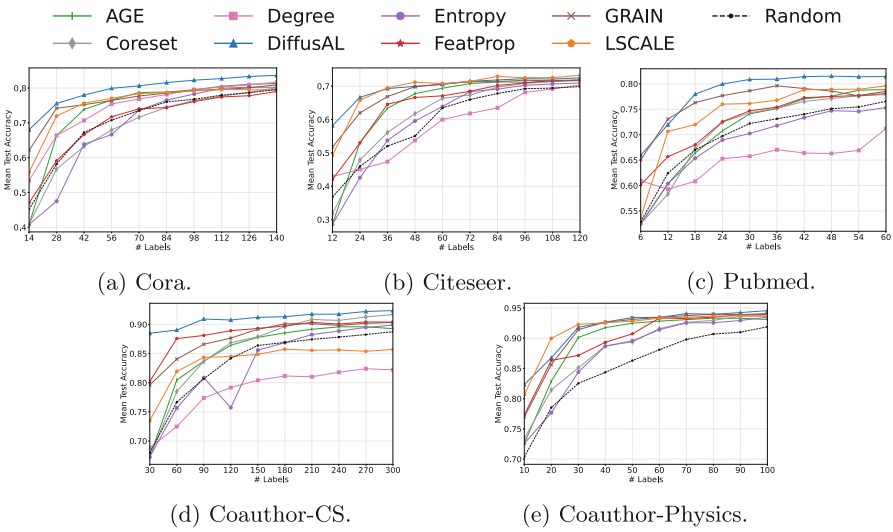
Implementation Details. All experiments were implemented using PyTorch [27] and PyTorch Geometric [13] and run on a single Nvidia Quadro RTX 8000 GPU. For more details, we refer to our publicly available codebase[1].

Competitors. We compare DiffusAL with *random* sampling, *entropy* sampling [32], and *coreset* [31] as graph-independent uncertainty-aware and diversity-aware active learning strategies, respectively. Furthermore, we include *degree* sampling as a graph-based representativeness-based baseline, selecting the highest degree nodes, as well as the state-of-the-art graph-specific active learning methods *AGE* [8], *FeatProp* [37], *LSCALE* [23] and *GRAIN* [43].

As proposed in [8,23,37,43], all baselines use GCNs as classifiers, except LSCALE, which uses the proposed distance-based classifier. Our proposed method DiffusAL uses the introduced QBC as a classifier, and we provide comprehensive experiments showing the influence of the prediction model.

Hyperparameters. We use the same hyper-parameters having a hidden layer size of 16, a dropout rate of 0.5, a learning rate of 0.01, and L2-regularization of 5×10^{-4} as proposed in [37]. For DiffusAL, we select α and ϵ as suggested in [7]. We follow a batch selection and retrain from scratch after each acquisition round. However, to ensure more diverse uncertainties (and because the other two scores are static), we follow the setting of [8] and also incrementally train the model for one epoch between instance selection within one acquisition round. The evaluation in Sect. 4.4 shows that this does not impair our efficiency. To provide a meaningful evaluation without the effects of an under-trained model or randomness factors, we report test accuracy for all approaches using a validation set of size 500 and early stopping. However, the validation set is only part of the evaluation, not the procedure itself. We set the size of the initial pool to 2C (cf. 3.1) and report results up to a budget of 20C with step sizes also twice the number of classes. To simulate a fairly realistic active learning scenario, the initial pool is sampled randomly without guaranteeing class balance for the baseline approaches without a specific initialization method. All experiments report an average of ten random seeds.

[1] https://github.com/lmu-dbs/diffusal.



(a) Cora. (b) Citeseer. (c) Pubmed.

(d) Coauthor-CS. (e) Coauthor-Physics.

Fig. 3. Active learning curves with the number of labeled nodes on the x-axis and average accuracy (over 10 random seeds) on the y-axis.

4.2 R1 - Performance Comparison

Figure 3 depicts the active learning curves for all budgets and datasets. DiffusAL (blue) is among the best-performing methods on all datasets. Especially on Cora and Coauthor-CS, we reach the highest mean accuracy for all labeling budgets and are the only competitor to reach a final accuracy of 83.6% and 92.4%, respectively. On Pubmed, GRAIN is similarly strong for the first two iterations. However, afterward, DiffusAL outperforms all methods for the remaining budgets and reaches a final average accuracy of 81.4%. In comparison, LSCALE, the second-best performing method with respect to the final budget, only reaches 79.9%.

On Citeseer and Physics[2], GRAIN and LSCALE are similarly strong as DiffusAL. For both datasets, the learning curves converge to similar accuracies above a certain labeling budget for some methods such that a clear winner can no longer be pronounced. Therefore, Fig. 4 provides a comprehensive dueling matrix indicating how often each strategy has won and lost against the other strategy in a similar fashion as was proposed in [1]. We apply a two-sided t-test with a p-value of 0.05 to the classification accuracies over 10 random seeds to count whether one method outperformed another with statistical significance.

[2] On Physics, Degree underperformed considerably and is therefore omitted for better presentation.

In total, we evaluated 50 experimental settings for each strategy (5 different datasets, 10 different labeling budgets from 2C to 20C). The values in a column and row of a method denote the percentage of losses and wins against another method, respectively. The bottom row indicates the average losses of each strategy over all experiments, and the right-most column indicates the average wins of a strategy over all experiments. The losses and wins in the cells c_{ij} and c_{ji} do not necessarily add up to 100%. The margin between the wins in cell ij and the losses in cell ji indicates how often the strategy i has performed equally well as competitor j. Both numbers, the average losses *and* the average wins, are particularly interesting when evaluating the success of an active learning method.

	DiffusAL	GRAIN	LSCALE	AGE	FeatProp	Coreset	Entropy	Random	Degree	Average Wins (%)
DiffusAL	**0**	**62**	**62**	**72**	**72**	**86**	**92**	**100**	**94**	71
GRAIN	**0**	0	26	36	42	58	72	90	80	45
LSCALE	**2**	4	0	28	34	54	66	72	78	38
AGE	**0**	0	14	0	20	34	48	58	68	27
FeatProp	**0**	4	22	14	0	16	42	44	62	23
Coreset	**0**	4	14	8	2	0	28	28	58	16
Entropy	**0**	0	8	0	6	4	0	18	46	9.1
Random	**0**	0	8	0	0	4	0	0	52	7.1
Degree	**0**	0	4	6	12	22	16	18	0	8.7
Average Losses (%)	0.2	8.2	18	18	21	31	40	48	60	

Fig. 4. Pairwise dueling matrix. Cell ij indicates how often competitor i won against competitor j **with statistical significance** over all datasets and labeling budgets (in %). The bottom-most row and right-most column denote each method's average losses and wins, respectively (in %).

In summary, the dueling matrix reveals the following insights:

- DiffusAL has the **fewest losses (0.2%, see first column)** and the **most wins (71%, see first row)**.
- DiffusAL **wins over random sampling most often (100%)**.
- Concerning wins over random sampling, GRAIN is the second-best method (90%). However, DiffusAL statistically never loses against GRAIN.
- The only strategy that can outperform DiffusAL is LSCALE. However, we beat LSCALE in 62% of experiments and lost only 2% of experiments.

4.3 R2 - Analysis of Contributing Factors

The selected datasets vary widely in terms of the number of nodes, edges, features, classes, and class distribution, making it difficult to develop an approach that can perform well across the spectrum. In the following, we analyze which components contribute most to DiffusAL's success and why it is so strong over a broad range of datasets. Table 2 shows the performance of DiffusAL (bottom row) and DiffusAL when switching off individual parts of the acquisition function, i.e., the diversity component (D), the uncertainty score (U) and the importance score (I) and exchanging the model architecture (middle rows) for 2C, 6C, and 12C labeling budgets on all datasets where C is the number of classes. Red, bold numbers indicate the smallest accuracy, indicating the largest influence of a switched-off component, and blue, bold numbers indicate the highest accuracy.

We exchange the classifier with a single network variant (MLP) and with a GCN taking the raw features as input instead of diffused features (GCN). Furthermore, we report results when using an additive score instead of a multiplicative score.

Table 2. Comparison of DiffusAL with ablated variants. **Blue, bold** numbers indicate the **highest, i.e. best,** accuracy. **Red, bold** numbers indicate the **lowest, i.e. worst,** accuracy and hence the component with largest influence.

			Cora			Citeseer			Pubmed			CS			Physics		
D	U	I	2C	6C	12C	2C	6C	12C	2C	6C	12C	2C	6C	12C	2C	6C	12C
		✓	45.5	77.8	80.6	**43.3**	63.8	69.7	56.3	68.0	69.8	71.9	81.7	82.2	71.9	89.8	93.8
	✓		45.5	76.1	80.1	43.3	65.5	70.0	56.3	70.6	75.4	71.9	83.3	90.8	71.9	89.6	93.1
	✓	✓	45.5	78.5	81.7	43.3	69.8	71.3	56.3	75.3	80.0	71.9	89.3	91.4	71.9	92.4	93.9
✓			-	74.5	76.0	-	67.6	71.1	-	64.6	76.5	-	89.4	90.4	-	86.4	87.1
✓		✓	-	76.4	80.5	-	67.7	71.0	-	77.2	79.9	-	87.5	87.3	-	91.5	92.4
✓	✓		-	78.6	81.9	-	69.1	71.0	-	74.9	77.1	-	90.5	91.6	-	88.3	90.9
Additive			-	78.8	81.3	-	**70.8**	71.3	-	79.1	80.2	-	**91.0**	**92.1**	-	91.7	92.7
MLP			62.0	78.8	81.8	52.7	70.6	**71.8**	64.1	78.8	79.9	87.8	90.4	91.2	80.4	91.4	93.6
GCN			61.8	77.5	80.7	49.8	69.3	71.3	64.5	76.5	78.5	83.3	89.6	91.2	**82.7**	91.6	93.1
DiffusAL			**68.0**	**79.9**	**82.3**	**58.2**	69.9	**71.8**	**65.9**	**80.0**	**81.4**	**88.5**	90.8	91.8	82.3	**92.6**	**94.1**

The importance, uncertainty, and additive scoring have no influence on the initial pool selection, so we leave out numbers there. Our QBC robustifies the accuracy, especially in the first iteration, compared to the other two variants (MLP, GCN). The performance difference between the models gets smaller with increasing label information. In particular, when label information is sparse, the committee stabilizes the prediction. However, the diversity component has the largest impact on the initial set for all datasets. When switching off diversity (first three rows), the accuracy drops between 9.6% (Pubmed) and 22.5% (Cora). Other approaches, such as FeatProp or LSCALE, also use clustering in the first iteration. However, our sampling directly operates on the diffused features, which subsequently directly influence the training and thus results in a very strong initial performance.

In general, switching off two scores yields worse results than only switching off one score, which indicates that the other two scores stabilize the results. But there is not one most important score over all datasets, supporting our claim that a robust selection benefits from diverse criteria. For instance, the accuracy drops the most when switching off *uncertainty* and *diversity* on Citeseer (by 2.1%) and especially on Coauthor-CS (by 9.6%). However, the performance on Cora and Physics primarily needs *uncertainty* and *importance*. In contrast, Pubmed benefits most from *diversity* and *importance*. Interestingly, some of our findings might give an indication of the performance of other methods. For instance, we found that importance, i.e., representativeness, is not beneficial on Coauthor-CS. LSCALE, which mainly focuses on representativeness sampling, yields the worst performance on this dataset. On Pubmed, however, uncertainty seems not to work well. Entropy and AGE both include uncertainty sampling and

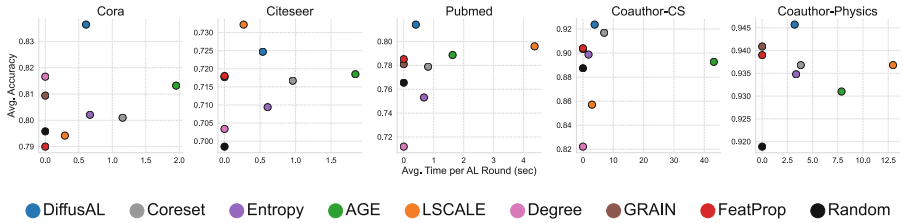


Fig. 5. Average time in seconds (x-axis) required for one active learning round compared to the average final accuracy (y-axis) for all methods (color). (Color figure online)

Table 3. Average time in seconds required for acquisition (acq), training (train), and in total ($\sum$) within one active learning iteration. Bold and underlined numbers indicate the fastest and second fastest methods, respectively. In total, DiffusAL is the fastest method on Physics and Pubmed, and the second fastest method on Cora and Citeseer.

	CS			Citeseer			Cora			Physics			Pubmed		
	acq	train	$\sum$	acq	train	$\sum$	acq	train	$\sum$	acq	train	$\sum$	acq	train	$\sum$
AGE	41.271	1.849	43.120	1.177	0.665	1.842	1.409	0.544	1.953	4.506	3.366	7.873	0.952	0.679	1.631
Coreset	5.191	1.797	6.988	0.344	0.615	0.960	0.537	0.616	1.154	0.572	3.258	3.830	0.138	0.675	0.813
Entropy	**0.005**	1.831	**1.836**	**0.002**	0.605	0.607	**0.002**	0.665	0.667	**0.011**	3.358	3.369	**0.002**	0.674	0.676
LSCALE	2.649	**0.317**	2.966	0.019	**0.249**	**0.269**	**0.042**	**0.250**	**0.292**	12.722	**0.258**	12.980	4.121	**0.247**	4.368
DiffusAL	2.567	1.282	3.849	0.183	0.356	0.539	0.268	0.339	0.608	0.357	2.863	**3.220**	0.043	0.361	**0.404**

yield worse results. On Cora, where uncertainty and representativeness seem effective, Coreset and FeatProp, which mainly focus on diversity, are among the worst-performing methods.

Using an *additive* score instead of a multiplicative score yields slightly worse results in general. From 10 comparisons, summing up the scores only yields three times slightly better results. However, the maximum difference is 0.9% (Citeseer 6C), whereas using the multiplicative in DiffusAL, the additive score is up to 1.4% (Physics 12C) better.

4.4 R3-Acquisition and Training Efficiency

Figure 5 shows the total average time (in seconds) for one active learning step on the x-axis (smaller is better) and the final accuracy after all 20C labels are selected on the y-axis (larger is better) for all methods (color).

We focus on an iterative AL selection where re-training between acquisition steps is necessary to get new uncertainty scores. In contrast, GRAIN, FeatProp, degree sampling, and random sampling select all instances for labeling at once and do not require re-training. Therefore, their average time is set to zero, and their accuracy is plotted for comparison. However, these methods are generally less label-efficient since they are not directly coupled to the current learning model. Except for Citeseer, DiffusAL is always on the Pareto-front, yielding the best final average accuracy while still being fairly time-efficient. In Table 3, we

split the total time into the acquisition and the training time for the iterative methods. All GCN-based methods (Coreset, AGE, Entropy) denote fairly similar training times. Despite using an ensemble, DiffusAL is slightly faster than the GCN-based methods since the features are pre-computed. AGE and Coreset both require a longer time for acquisition. AGE can exploit pre-calculated centrality scores. However, the uncertainty score and especially the density score must be freshly calculated in each round. Especially for the very large graph data CS, AGE requires over 40 s for one active learning iteration. Coreset extracts the latent representations from the current model and requires the computation of a pairwise distance matrix. Compared to that, DiffusAL only needs to calculate the uncertainty scores derived from the QBC model since the other scores are pre-computed. Only the entropy-based selection scheme has a faster acquisition time since it only needs one forward pass through the network.

LSCALE, which also defined a dedicated network towards a unified learning and selection framework, has the fastest training times out of all methods. However, depending on the dataset, the acquisition time is much larger than DiffusAL's acquisition time. As such, the overall time needed for one active learning round varies considerably between datasets. For instance, on Citeseer and Cora, LSCALE is the fastest method out of all iterative methods. Still, on the much larger graphs Pubmed and Physics, it is the slowest method due to larger acquisition times (4.4 s and 12.7 s, respectively). Overall, even though we use an ensemble method, our training and acquisition times are fairly stable across datasets and, in total, comparably good as plain uncertainty sampling with a GCN.

5 Conclusion

The annotation of unlabeled nodes in graphs is a time-consuming and costly task and, accordingly, it is of great interest to advance label-efficient methods. Motivated by the success of diffusion-based graph learning approaches, we propose DiffusAL, a novel active learning strategy for node classification. DiffusAL uses diffusion to predict node labels accurately and compute meaningful utility scores consisting of *model uncertainty*, *diffused feature diversity*, and *node importance* for active node selection, such that training and data selection cooperate toward label-efficient node classification. DiffusAL is significantly better generalizable over a wide range of datasets and is, in terms of statistical significance, not beaten by any other method in 99.8% of all experiments. Moreover, it is the only method that significantly outperforms random selection in 100% of the evaluated settings. Due to pre-computed features stored in a diffusion matrix, our model can efficiently compute a node's utility for training and acquisition. Our extensive ablation study shows that each component of DiffusAL contributes to different datasets and active learning stages, making it robust in diverse graph settings.

References

1. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep batch active learning by diverse, uncertain gradient lower bounds. In: ICLR (2020)
2. Battaglia, P.W., et al.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018)
3. Bilgic, M., Mihalkova, L., Getoor, L.: Active learning for networked data. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 79–86 (2010)
4. Borutta, F., Busch, J., Faerman, E., Klink, A., Schubert, M.: Structural graph representations based on multiscale local network topologies. In: WI-IAT (2019)
5. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE SPM **34**(4), 18–42 (2017)
6. Busch, J., Kocheturov, A., Tresp, V., Seidl, T.: Nf-gnn: network flow graph neural networks for malware detection and classification. In: SSDBM (2021)
7. Busch, J., Pi, J., Seidl, T.: Pushnet: efficient and adaptive neural message passing. In: ECAI (2020)
8. Cai, H., Zheng, V.W., Chang, K.C.C.: Active learning for graph embedding. arXiv preprint arXiv:1705.05085 (2017)
9. Chandra, A.L., Desai, S.V., Devaguptapu, C., Balasubramanian, V.N.: On initial pools for deep active learning. In: NeurIPS 2020 Workshop on Pre-registration in Machine Learning, pp. 14–32. PMLR (2021)
10. Contardo, G., Denoyer, L., Artières, T.: A meta-learning approach to one-step active-learning. In: AutoML@PKDD/ECML (2017)
11. Faerman, E., Borutta, F., Busch, J., Schubert, M.: Semi-supervised learning on graphs based on local label distributions. In: MLG (2018)
12. Faerman, E., Borutta, F., Busch, J., Schubert, M.: Ada-lld: adaptive node similarity using multi-scale local label distributions. In: WI-IAT (2020)
13. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
14. Frey, C.M.M., Ma, Y., Schubert, M.: Sea: graph shell attention in graph neural networks. In: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2022)
15. Gao, L., Yang, H., Zhou, C., Wu, J., Pan, S., Hu, Y.: Active discriminative network representation learning. In: IJCAI (2018)
16. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML, pp. 1263–1272. PMLR (2017)
17. Hamilton, W.L.: Graph representation learning. Synth. Lect. Artifi. Intell. Mach. Learn. **14**(3), 1–159 (2020)
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
19. Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: graph neural networks meet personalized pagerank. In: ICLR (2019)
20. Klicpera, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. Adv. Neural. Inf. Process. Syst. **32**, 13354–13366 (2019)
21. Lee, J.B., Rossi, R., Kong, X.: Graph classification using structural attention. In: KDD, pp. 1666–1674 (2018)
22. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI (2018)

23. Liu, J., Wang, Y., Hooi, B., Yang, R., Xiao, X.: Lscale: latent space clustering-based active learning for node classification. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, 19–23 September 2022, Proceedings, Part I, pp. 55–70. Springer (2023). https://doi.org/10.1007/978-3-031-26387-3_4
24. Moore, C., Yan, X., Zhu, Y., Rouquier, J.B., Lane, T.: Active learning for node classification in assortative and disassortative networks. In: Proceedings of the 17th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, pp. 841–849 (2011)
25. Namata, G.M., London, B., Getoor, L., Huang, B.: Query-driven active surveying for collective classification. In: MLG (2012)
26. Ogawa, Y., Maekawa, S., Sasaki, Y., Fujiwara, Y., Onizuka, M.: Adaptive node embedding propagation for semi-supervised classification. In: Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., Lozano, J.A. (eds.) ECML PKDD 2021. LNCS (LNAI), vol. 12976, pp. 417–433. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86520-7_26
27. Paszke, A., et al.: Pytorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
28. Regol, F., Pal, S., Zhang, Y., Coates, M.: Active learning on attributed graphs via graph cognizant logistic regression and preemptive query generation. In: ICML, pp. 8041–8050. PMLR (2020)
29. Regol, F., Pal, S., Zhang, Y., Coates, M.: Active learning on attributed graphs via graph cognizant logistic regression and preemptive query generation. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, JMLR.org (2020)
30. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93 (2008)
31. Sener, O., Savarese, S.: Active learning for convolutional neural networks: a core-set approach. In: ICLR (2018)
32. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison (2009)
33. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT 1992, pp. 287–294. Association for Computing Machinery, New York (1992)
34. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. In: NeurIPS Relational Representation Learning Workshop (2018)
35. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
36. Veličković, P., Fedus, W., Hamilton, W.L., Lió, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: ICLR (2018)
37. Wu, Y., Xu, Y., Singh, A., Yang, Y., Dubrawski, A.: Active learning for graph neural networks via node feature propagation. arXiv preprint arXiv:1910.07567 (2019)
38. Wu, Y., Xu, Y., Singh, A., Yang, Y., Dubrawski, A.: Active learning for graph neural networks via node feature propagation. CoRR abs/ arXiv: 1910.07567 (2019)
39. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks?. In: ICLR (2019)

40. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: ICML, pp. 5453–5462. PMLR (2018)
41. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: Advances in Neural Information Processing Systems 31 (2018)
42. Zhang, W., Shen, Y., Li, Y., Chen, L., Yang, Z., Cui, B.: Alg: fast and accurate active learning framework for graph convolutional networks. In: SIGMOD, pp. 2366–2374 (2021)
43. Zhang, W., et al.: Grain: Improving data efficiency of graph neural networks via diversified influence maximization. Proc. VLDB Endow. **14**(11), 2473–2482 (2021)
44. Zhao, L., et al.: T-gcn: a temporal graph convolutional network for traffic prediction. IEEE Trans. Intell. Transp. Syst. **21**(9), 3848–3858 (2019)