Generative Flow Networks for Precise Reward-Oriented Active Learning on Graphs

Yinchuan Li[1], **Zhigang Li**[2], **Wenqian Li**[3], **Yunfeng Shao**[1], **Yan Zheng**[2], **Jianye Hao**[1,2*]
[1]Huawei Noah's Ark Lab
[2]Tianjin University
[3]National University of Singapore
{liyinchuan, shaoyunfeng}@huawei.com, {scs_lzg, yanzheng, sjianye.hao}@tju.edu.cn,
wenqian@u.nus.edu

Abstract

Many score-based active learning methods have been successfully applied to graph-structured data, aiming to reduce the number of labels and achieve better performance of graph neural networks based on predefined score functions. However, these algorithms struggle to learn policy distributions that are proportional to rewards and have limited exploration capabilities. In this paper, we innovatively formulate the graph active learning problem as a generative process, named GFlowGNN, which generates various samples through sequential actions with probabilities precisely proportional to a predefined reward function. Furthermore, we propose the concept of flow nodes and flow features to efficiently model graphs as flows based on generative flow networks, where the policy network is trained with specially designed rewards. Extensive experiments on real datasets show that the proposed approach has good exploration capability and transferability, outperforming various state-of-the-art methods.

1 Introduction

Graph neural networks (GNNs) have achieved great success in processing graph-structured data in recent years [Wu *et al.*, 2020b; Liu *et al.*, 2020; Liu *et al.*, 2021; Sarlin *et al.*, 2020; You *et al.*, 2020; Ying *et al.*, 2019; Hu *et al.*, 2020b]. It can simultaneously model structural information and extract node attributes [Jin *et al.*, 2020]. They provide significant performance improvements for graph-related downstream tasks such as node classification, link prediction, group detection, and graph classification [Wu *et al.*, 2020b; Liu *et al.*, 2019; You *et al.*, 2019; Zhu *et al.*, 2020]. Nevertheless, graph neural networks usually require a large amount of labeled data for training, which is expensive for some fields, such as chemistry and biomolecular design [Garg *et al.*, 2020]. Improving the utilization of graph neural networks for labeled data has become a key challenge.

Active learning applied to graphs is used to address this challenge. Graph active learning methods [Ma *et al.*, 2022; Hao *et al.*, 2020; Zhang *et al.*, 2021] usually select the most informative node set for labeling and input it to the classification graph neural network for training. The quality of the selected nodes directly determines the training effect of the classification graph neural network for the same number of label training sets. Previous methods usually select the label node set through heuristic criteria, such as degree, entropy, and distance from the central node [Gao *et al.*, 2018b; Gu *et al.*, 2013; Ji and Han, 2012]. There are also ways to find the optimal policies by maximizing information or maximizing cumulative reward [Hu *et al.*, 2020a]. However, these methods ignore that the probability of selecting a node should be consistent with the reward distribution it ultimately brings.

In this paper, we propose a novel active learning algorithm for GNNs that addresses this problem by exploiting the advantages of generative flow networks (GFlowNets) [Bengio *et al.*, 2021b], which has been applied in causal structure learning [Li *et al.*, 2022; Nishikawa-Toomey *et al.*, 2022], GNN explanations [Li *et al.*, 2023a], biological sequence design [Jain *et al.*, 2022], discrete probabilistic modeling [Zhang *et al.*, 2022a] and has been extended to continuous space [Li *et al.*, 2023c; Lahlou *et al.*, 2023; Li *et al.*, 2023b]. Unlike traditional approaches, we convert the sampling procedure in active learning into a generation problem, gradually adding nodes to the label set until the desired label set is generated. We name this method `GFlowGNN`, which generates various samples through sequential actions with probabilities precisely proportional to a predefined reward function. In particular, in GNN-based active learning tasks, the goal is to interactively select a sequence of nodes to maximize the performance of the GNN trained on them. We can model this problem as an MDP, where the current graph is the state, and the active learning system takes action by selecting nodes to query. We then obtain rewards by evaluating the performance of the trained GNN model with the labeled set.

By making the probability of node selection consistent with the node's reward distribution, GFlowGNN can better learn the optimal node selection policy and help avoid getting stuck in local optima. In addition, our GFlowGNN has better exploration capability than reinforcement learning (RL)-based methods [Hu *et al.*, 2020a] and can achieve faster convergence and better performance, since RL prefers to explore near the maximum reward while GFlowGNN explores according to the precise reward distribution.

*Corresponding author.
The codes are available at https://gitee.com/mindspore/models/tree/master/research/gflownets/gflowgnn

1.1 Main Contributions

In general, our contributions are mainly reflected in the following aspects: 1) we innovatively formulate the graph active learning problem as a generative process rather than a searching or optimization problem; 2) We propose the definition of GFlowGNN, a general architecture with unlimited types of reward functions and neural networks for solving graph active learning problems; 3) We propose two concepts of flow nodes and flow features to effectively model the graph state transition process as a flow model, where the flow model indirectly establishes a mapping relationship with the state through flow nodes and flow features. This partial update approach makes it possible to efficiently train policy networks in GFlowGNN; 4) Our GFlowGNN has good generalization and transferability, and extensive experimental results show that the proposed method outperforms various existing state-of-the-art methods on public datasets.

2 Related Work

2.1 Graph Neural Network

GNNs are deep learning based approaches that operate on graph domain, which are proposed to collectively aggregate information from graph structure [Zhou *et al.*, 2020]. Some subsequent research advance it further, for example, GCN [Kipf and Welling, 2016] presents a scalable approach for semi-supervised learning based on an efficient variant of convolutional neural networks, and GAT [Veličković *et al.*, 2017] proposes to apply the attention mechanism in the aggregation process. However, GNNs typically require a massive number of labeled data for training and the causes high annotation cost [Hu *et al.*, 2020a]

2.2 Active Learning

AL improves labeling efficiency by identifying the most valuable samples to label. There are various approaches such as Uncertainty Sampling [Yang *et al.*, 2015], ensembles [Zhang *et al.*, 2020] and Query-by-Committee [Burbidge *et al.*, 2007; Melville and Mooney, 2004]. A flurry of developments could be divided into Density-based [Zhu *et al.*, 2008; Tang *et al.*, 2002], Clustering-based [Du *et al.*, 2015; Nguyen and Smeulders, 2004] and Diversity-based [Wu *et al.*, 2020a; Jiang and Gupta, 2021] methods.

2.3 GNN Based Active Learning

Despite the effectiveness of common AL methods, it is unsuitable to directly apply them to GNNs since the characteristic of influence propagation has not been considered. To tackle this issue, both AGE [Cai *et al.*, 2017] and ANRMAB [Gao *et al.*, 2018b] introduce the density of node embedding and PageRank centrality into the node selection criterion. Similarly, ActiveHNE [Chen *et al.*, 2019] tackles active learning on heterogeneous graphs by posing it as a multi-armed bandit problem. SEAL [Li *et al.*, 2020] devises a novel AL query strategy in an adversarial way, and RIM [Zhang *et al.*, 2021] considers the noisy oracle in the node labeling process of graph data. Recently, IGP [Zhang *et al.*, 2022b] proposes the relaxed queries and soft labels to tackle the high cost problem in the exact labeling task.

3 GFlowGNN: Problem Formulation

Considering a directed graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_1, ..., v_n\}$ being a finite set of nodes, and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ representing directed edges. The graph is modeled by the binary adjacency matrix $A \in \{0,1\}^{n \times n}$. Nodes can be paired with features $\mathcal{X} = \{x_v | \forall v \in \mathcal{V}\} \subset \mathbb{R}^d$, and labels $\mathcal{Y} = \{y_v | \forall v \in \mathcal{V}\} = \{1, 2, ..., C\}$. The node set is divided into three subsets as $\mathcal{V}_{\text{train}}$, $\mathcal{V}_{\text{valid}}$ and $\mathcal{V}_{\text{test}}$.

Suppose the query budget is b, where $b \ll |\mathcal{V}_{\text{train}}|$. We initialize an empty labeled node set $\mathcal{V}^0_{\text{label}} = \emptyset$. Starting from $\mathcal{V}^0_{\text{label}}$, at each step t, we query an unlabeled node v_{t+1} from $\mathcal{V}_{\text{train}} \backslash \mathcal{V}^t_{\text{label}}$ based on the active learning policy π, and then update the labeled node set $\mathcal{V}^{t+1}_{\text{label}} = \mathcal{V}^t_{\text{label}} \cup \{v_{t+1}\}$. The classification GNN $f_t(\cdot)$ related with G is trained with the updated $\mathcal{V}^{t+1}_{\text{label}}$ for one more epoch. When the budget b is used up, we stop the query process and continue training the classification GNN $f_b(\cdot)$ with $\mathcal{V}^b_{\text{label}}$ until convergence. At each step t, the labeled set $\mathcal{V}^t_{\text{label}}$ is evaluated based on $\mathcal{M}(f_t(\mathcal{V}_{\text{valid}}))$, where $\mathcal{M}(\cdot)$ is the metric to evaluate the performance of the classification GNN $f(\cdot)$ trained based on $\mathcal{V}^t_{\text{label}}$ on the validation set $\mathcal{V}_{\text{valid}}$.

Previous approaches aim to find the optimal active learning policy $\pi^\star$ that can sample a label set $\mathcal{V}^\pi_{\text{valid}}$ satisfying

$$\pi^\star = \arg\max_\pi \mathcal{M}(f_\pi(\mathcal{V}_{\text{valid}})). \tag{1}$$

For each graph, we have no labels initially, then we select a sequence of nodes to label based on $\pi^\star$ and use them to train the classification GNN on G.

Obviously, the performance of GNN directly depends on the performance of the active learning policy π. Heuristic methods [Gao *et al.*, 2018b; Gu *et al.*, 2013; Ji and Han, 2012] are based on degree, entropy, and distance to obtain better policies, which are not necessarily optimal. Reinforcement learning-based methods [Hu *et al.*, 2020a] can use the accuracy of GNNs as a metric to learn a better policy. However, the learned policy may not be precisely consistent with the accuracy distribution, resulting in performance loss.

In this paper, we consider this labeling process as the generation process, in which $\mathcal{V}^b_{\text{label}}$ is a compositional object. Starting from an empty set, we can use a neural network as a sampler to generate such $\mathcal{V}^b_{\text{label}}$ by sequentially querying a new label each time. After the query budget b is used up, we use it to train GNN classification f and use its performance score on the validation set as the reward $r(\mathcal{V}^b_{\text{label}})$ to evaluate this label set. Unlike traditional methods having maximization objectives, our goal is to learn a forward generative policy for active learning that could sample $\mathcal{V}^b_{\text{label}}$ with probabilities highly consistent with the reward distribution (metric distribution), so that GNNs perform better. Therefore, our GFlowGNN is defined as following Definition 1.

Definition 1 (GFlowGNN). *Given a directed graph G and a reward function $r(\cdot)$, GFlowGNN aims to find the best forward generative policy π to sample a sequence of nodes based on generative flow networks, such that*

$$\pi(\mathcal{V}^b_{label}; \theta) \propto r(\mathcal{V}^b_{label}), \tag{2}$$

where θ is the parameter of flow network, and π can be transferred to other datasets.

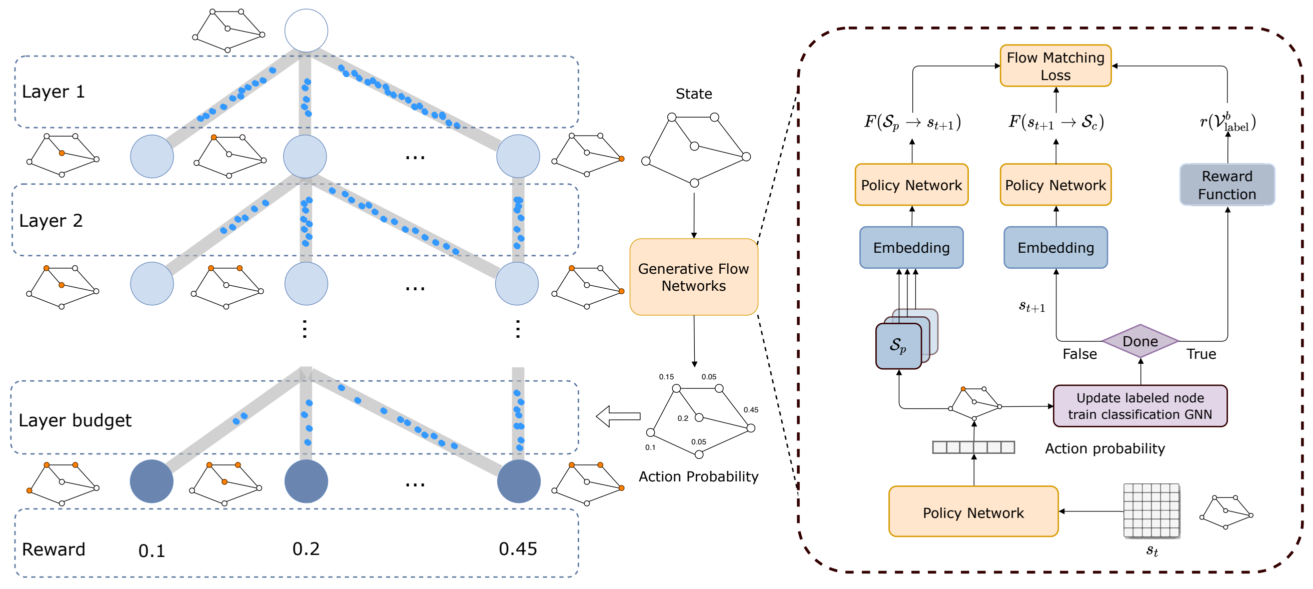

Figure 1: The framework of GFlowNets-based active learning on GNNs aims to train the policy that action probabilities follow true reward distribution. We model the MDP as a directed acyclic graph on the left side. A circle node corresponds to a state, and the edge between two nodes corresponds to the state transition. The initial state is the node in white, the interior states are the nodes in light blue while final states are the nodes in dark blue. Given a state, the policy network calculates the unnormalized action probability in each layer, corresponding to the blue dots on the grey edge, which we could consider as the water pipe. The number of blue dots represents the water flow, equivalent to an action's probability. After the layer budget, each label set is evaluated based on the reward function. The figure on the right shows the training process of the policy network.

4 GFlowGNN: Framework

4.1 Overall Framework

We show an overview of the policy training framework in Figure 1. We initialize a new empty set, which corresponds to the initial graph with all hollow nodes above the layer 1 in the left side. Each step the generative flow network computes the edge flow, which corresponds to the blue dots in the grey pipe, as unnormalised action probabilities based on the current input state. Then sample an unlabeled node and add it for the label node set, which is consistent with the nodes highlighted in orange in a graph. The layer i refers to the i-th state transition in a trajectory, and this process iterates until the query budget b is used up. When a $\mathcal{V}^b_{\text{label}}$ is generated, we use it to train a GNN classification model f and use the model performance as the reward of this label set. In the GFlowNets (the right side), the unlabeled node v_t is sampled based on the action probability calculated by the NNs and the f is trained based on the updated label set. After completing the trajectories, the policy net is trained based on the flow matching loss calculated by the inflow, outflow or rewards of the states. We will show more details about the flow network architecture in the the later section.

4.2 Modeling Graphs as Flows

To ensure the learned policy could sample compositional objects proportional to the reward, the concept of "flow" is naturally introduced to model our policy. However, modeling graphs as flows is not easy. We need to construct suitable states and actions to ensure that a directed acyclic graph (DAG) (with state sets $\mathcal{S}$ and action sets $\mathcal{A}$) can be formed, where acyclic means that there is no trajectory $(s_0, s_1, ..., s_f)$ satisfying $s_i = s_j, \forall i \neq j$.

To form DAG, we propose two concepts of *flow node* and *flow feature* defined as follows:

Definition 2 (Flow Node)**.** *Considering the flow through the entire DAG, a flow node is defined as a node in the DAG. Once a flow node is determined, its corresponding child and parent nodes can be determined.*

According to Definition 2, the flow node is mainly used to determine the parent node and child node, which can further ensure the establishment of the acyclic property in the DAG. To model a graph as a flow, we can construct a *node vector* to satisfy the properties of flow nodes. In particular, we define an indicator vector to represent whether a node has been labeled or not, i.e.,

$$\mathbf{v}_i(s_t) = \mathbb{1}\{v_i \in \mathcal{V}^t_{\text{label}}\},\ i = 1, ..., n. \tag{3}$$

For the initial state s_0, its node vector is an all-zero vector corresponding to the empty label set. In this way, the flow nodes in the DAG structure can be transformed by the action proposed in Definition 3.

Definition 3 (Action)**.** *At time step t, an action $a_t : s_t \to s_{t+1} \in \mathcal{A}$ is to select a node from $\mathcal{V}_{train} \backslash \mathcal{V}^t_{label}$, i.e. if node v_i is selected, it is the same as assigning position $\mathbf{v}_i = 1$ in $\mathbf{v}(s_t)$.*

Flow nodes can help construct effective DAG graphs, but for complex GNN tasks, it is difficult to calculate the flow on nodes only by flow node vectors, and cannot be strictly reward-oriented. To solve this problem, we propose the concept of flow features in Definition 4.

Definition 4 (Flow Feature)**.** *Flow features are defined as detailed features associated with flow nodes to aid in computing more accurate flow and policy.*

Note that the flow feature is not used to find the parent node and child node, i.e., not used to determine the DAG structure. The flow feature here is defined as a matrix $\mathbf{\Phi}(s_t) \in \mathbb{R}^{n\times 4}$. In particular, the first column of $\mathbf{\Phi}$ denotes the degree of n nodes in the graph. Denote $\mathcal{N}(v)$ as the set of all neighbor nodes of v, the degree is calculated by

$$\mathbf{\Phi}_{i,1}(s_t) = \min(|\mathcal{N}(v_i)|/\alpha, 1),\ i = 1, ..., n, \tag{4}$$

which is scaled by the hyperparameter α and clipped to 1.

The second column of $\mathbf{\Phi}$ denotes the uncertainty of n nodes in the graph, which is calculated by the entropy of the label probabilities $\hat{y} \in \mathbb{R}^C$ predicted by the GNN, i.e.,

$$\mathbf{\Phi}_{i,2}(s_t) = H(\hat{y}^t(v_i))/\log(C),\ i = 1, ..., n. \tag{5}$$

The third and fourth columns of $\mathbf{\Phi}$ denote the divergence of n nodes in the graph, which are calculated based on a node's predicted label probabilities and its neighbor's, i.e.,

$$\mathbf{\Phi}_{i,3}(s_t) = \frac{1}{|\mathcal{N}(v_i)|} \sum_{u\in\mathcal{N}(v_i)} \mathrm{KL}(\hat{y}^t(v_i)\|\hat{y}^t(u)), \tag{6}$$

$$\mathbf{\Phi}_{i,4}(s_t) = \frac{1}{|\mathcal{N}(v_i)|} \sum_{u\in\mathcal{N}(v_i)} \mathrm{KL}(\hat{y}^t(u)\|\hat{y}^t(v_i)) \tag{7}$$

for $i = 1, ..., n$. Note that the flow features used here are flexible. These common features are recognized as informative and effective in graph structured data on both theoretical and experimental results. Therefore, we choose them as flow features in our paper, which keeping in line with other baselines.

4.3 How to Find Parent Nodes

The process of exploring the parent nodes is tied to changing the value in the node vector $\mathbf{v}$, for example, converting the value in $\mathbf{v}$ from 1 to 0. Given any $\mathbf{v}$, if $\|\mathbf{v}\|_0 = m$, which means this state has m different parent states, then one of its parents is obtained by converting one of the elements with value 1 to value 0 in $\mathbf{v}$ to generate a new vector $\mathbf{v}'$.

As for the flow features $\mathbf{\Phi}$ of the parent nodes, computing all parent states' dynamic flow features is time-consuming and impractical. We hence directly store all the flow features $\mathbf{\Phi}$ for each round, and directly use the flow feature of s_{t-1} as the parent flow features of s_t, i.e.,

$$\mathbf{\Phi}(\text{parent}[s_t]) \leftarrow \mathbf{\Phi}(s_{t-1}).$$

This approach seems to be empirical; however, through extensive experimental studies, we found that this approach does not sacrifice much performance but significantly improves the efficiency of the algorithm.

4.4 Flow Modeling

Given above DAG, our flow here is defined as a non-negative function $F_{\mathbf{\Phi},\mathbf{v}}(s_t)$ related the flow node $\mathbf{v}$ and flow features $\mathbf{\Phi}$, which has an interesting difference from the definition in [Bengio *et al.*, 2021b].

Remark 1. *In [Bengio* et al.*, 2021b], the flow model is only related to the state of node, and the state must be fully updated to calculate the flow model. In reality, many useful features cannot be strictly updated, or the computational load is large. Our flow model indirectly establishes a mapping relationship with the state through flow nodes and flow features, in which the flow nodes have a strict one-to-one correspondence and need to be updated accurately. In contrast, the flow feature provides more details to assist the calculation of the flow model, and does not need to be updated accurately every round, e.g., when calculating the flow features of parent nodes.*

Define $\mathcal{T}$ as the set of complete trajectories related to the given DAG, a *complete trajectory* is defined as a sequence of states $\tau = (s_0, ..., s_f) \in \mathcal{T}$ with s_0 being the initial state and s_f being the final state. Define an *edge flow* or *action flow* $F_{\mathbf{\Phi},\mathbf{v}}(s_t \to s_{t+1}) = F_{\mathbf{\Phi},\mathbf{v}}(s_t, a_t) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ as the flow through an edge $s_t \to s_{t+1}$, where $a_t : s_t \to s_{t+1}$. The *state flow* $F_{\mathbf{\Phi},\mathbf{v}}(s) : \mathcal{S} \mapsto \mathbb{R}$ is defined as the sum of the flows of the complete trajectories passing through it, i.e.,

$$F_{\mathbf{\Phi},\mathbf{v}}(s) = \sum_{\tau\in\mathcal{T}} \mathbb{1}_{s\in\tau} F_{\mathbf{\Phi},\mathbf{v}}(\tau).$$

In the flow model, the probability of each action can be easily measured by the fraction of action flow $F_{\mathbf{\Phi},\mathbf{v}}(s, a)$ to state flow $F_{\mathbf{\Phi},\mathbf{v}}(s)$. The property of the flow model is crucial to achieve precise proportionality between policies and rewards, which will be introduced in detail later. By exploiting the advantage of flow model, our GFlowGNN has good generalization and transferability, and extensive experimental results show that the proposed method outperforms various existing state-of-the-art methods on public datasets.

4.5 Reward Design

We first analyze how the flow model contributes to establish a policy that could sample the final action exactly proportional to rewards. Then we design the reward explicitly based on the advantage of this flow model.

For any flow function $F_{\mathbf{\Phi},\mathbf{v}}(s_t)$, the forward transition probability $\mathcal{P}_F(s_t \to s_{t+1}|s_t)$ is given by [Bengio *et al.*, 2021b]

$$\mathcal{P}_F(s_{t+1}|s_t) := \mathcal{P}_F(s_t \to s_{t+1}|s_t) = \frac{F_{\mathbf{\Phi},\mathbf{v}}(s_t \to s_{t+1})}{F_{\mathbf{\Phi},\mathbf{v}}(s_t)}. \tag{8}$$

Then, we have $\mathcal{P}_F(\tau) = \prod_{t=0}^{f-1} \mathcal{P}_F(s_{t+1}|s_t)$, which yields

$$\mathcal{P}_F(s_f) = \frac{\sum_{\tau\in\mathcal{T}} \mathbb{1}_{s_f\in\tau} F_{\mathbf{\Phi},\mathbf{v}}(\tau)}{\sum_{\tau\in\mathcal{T}} F_{\mathbf{\Phi},\mathbf{v}}(\tau)} = \sum_{\tau\in\mathcal{T}} \mathbb{1}_{s_f\in\tau} \mathcal{P}_F(\tau) \tag{9}$$

by noting that $F_{\mathbf{\Phi},\mathbf{v}}(s_f) = \sum_{\tau\in\mathcal{T}} \mathbb{1}_{s_f\in\tau} F_{\mathbf{\Phi},\mathbf{v}}(\tau)$. Then we have

$$\mathcal{P}_F(s_f) \propto F_{\mathbf{\Phi},\mathbf{v}}(s_f), \tag{10}$$

where $F_{\mathbf{\Phi},\mathbf{v}}(s_f) = r(s_f)$.

Let $\pi : \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ be the probability distribution $\pi(a|s)$ over actions $a \in \mathcal{A}$ for each state $s \in \mathcal{S}$, which first appears in Definition 1. We can map the policy π to the flow-based transition probabilities based on the flow properties,

$$\pi(\mathcal{V}^b_{\text{label}}) = \prod_{t=0}^{f-1} \pi(a_t|s_t) = \mathcal{P}_F(s_f). \tag{11}$$

By combining (10) and (11), we can obtain (2) in Definition 1, i.e.,

$$\pi(\mathcal{V}_{\text{label}}^b;\theta) \propto r(\mathcal{V}_{\text{label}}^b),$$

which reveals that the learned policy is proportional to the reward.

Since the policy approximately samples proportionally to the reward, we can explicitly design the rewards as model accuracy, thus ensuring the sampling diversity and model performance. Formally, given a sequence of labeled nodes $\mathcal{V}_{\text{label}}^b = (v_1, ..., v_b)$, we define the trajectory reward as

$$r(\mathcal{V}_{\text{label}}^b) = \mathcal{M}(f_b(\mathcal{V}_{\text{valid}})), \tag{12}$$

where the evaluation metric is the prediction accuracy of f_b.

4.6 Policy Network Architecture

The role of the policy network is to calculate the probability score of each node based on the input state matrix Φ, that is, to learn the policy π. In this paper, we propose two policy network architectures for non-transfer scenarios and transfer scenarios, respectively.

For non-transfer scenarios, i.e., the training and test datasets are i.i.d. and the data has the same dimension, we use MLP as our policy network. We feed the state matrix consisting of the features of each node into the policy network after flatten embedding, and then obtain the probability score of each node. The MLP is set as four layers, where the input dimension is the number of nodes multiplied by the node feature dimension, while the output is the probability of each node to be labeled, whose dimension equals the number of nodes.

For transfer scenarios, the training and test datasets are different, our policy network mainly based on the Graph Convolutional Network (GCN) [Zhang *et al.*, 2019] to obtain the generalizable property. GCN starts from node attributes, update node embeddings through several layers of neighborhood aggregation, and then calculate edge and graph embeddings from updated node embedding. The node representation of GCN is updated by:

$$H_{\ell+1} = \sigma(D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}H_\ell W_\ell),\ \ell = 1, ..., L-1, \tag{13}$$

where H_ℓ and W_ℓ are respectively the input feature matrices and the weight of layer ℓ; $H_0 = \Phi(s_t)$; I is an identity matrix such that $A + I$ denotes the adjacency matrix with self loops; D is a diagonal matrix with $D_{i,i} = \sum_j A_{i,j}$; $\sigma(\cdot)$ denotes the ReLU activation function.

Once H_L is available, we apply a linear layer to obtain the predicted probability by

$$p_t = \text{Softmax}(H_L w + b), \tag{14}$$

where $\text{Softmax}(\cdot)$ denotes the softmax function, w denotes the weights of the linear layer and b denotes the bias. It is noteworthy to note that our neural network is designed to approximate the flow, i.e., to obtain $\hat{F}(s, a)$. Therefore, to get the predicted probability, we need to add a Softmax operator for normalization, i.e. if we fix the total flow to 1, we can get the transition probability by directly normalizing the action flow.

5 GFlowGNN: Training Procedure

5.1 State Transition Dynamics

At each step, the selected node v_t will be labeled and then be added to $\mathcal{V}_{\text{label}}$ to update the classification GNN f. Then based on the predictions of GNN, we update the state matrix especially dynamic features. All parents $s \in \mathcal{S}_p(s_{t+1})$ of s_{t+1} are explored after making a transition from s_t to s_{t+1}, where $\mathcal{S}_p(s_{t+1})$ indicates the parent set of s_{t+1}. Then the inflows $F_{\mathbf{\Phi},\mathbf{v}}(s \to s_{t+1})|_{s\in\mathcal{S}_p(s_{t+1})}$ and the outflows $F_{\mathbf{\Phi},\mathbf{v}}(s_{t+1} \to s)|_{s\in\mathcal{S}_c(s_{t+1})}$ are calculated, where $\mathcal{S}_c(s_{t+1})$ is the child set of s_{t+1}. When the budget is used up, the agent calculates the reward $r(s_f)$ instead of the outflow.

5.2 Flow Matching Loss

GFlowGNN generates complete trajectories $(s_0, s_1, ..., s_f) \in \mathcal{T}$ by iteratively sampling $v_t \sim \pi(a_t|s_t)$ starting from an empty label set until the budget is depleted. After sampling a buffer, we use the flow matching loss proposed by [Bengio *et al.*, 2021a] to train the policy $\pi(a_t|s_t)$ which satisfies $\mathcal{P}_F^\theta(s_f) \propto r(\mathcal{V}_{\text{label}}^b) = r(s_f)$,

$$\mathcal{L}_p(\tau) = \sum_{s_{t+1}\in\tau\neq s_0} \left\{ \log\left[\epsilon + \sum_{s_t,a_t:T(s_t,a_t)=s_{t+1}} F_t'\right] \right. \tag{15}$$
$$\left. - \log\left[\epsilon + \mathbb{1}_{s_{t+1}=s_f} r(s_{t+1}) + \mathbb{1}_{s_{t+1}\neq s_f} \sum_{a_{t+1}\in\mathcal{A}(s')} F_{t+1}'\right]\right\}^2,$$

where $F_t' = \exp(\log F_\theta(s_t, a_t))$ with θ being the network parameters. $\sum_{s_t,a_t:T(s_t,a_t)=s_{t+1}} F_t'$ indicates the inflow of s_{t+1}, i.e., the sum of edge/action flows from the parent states set $\mathcal{S}_p(s_{t+1})$, where $a_t : T(s_t, a_t) = s_{t+1}$ indicates an action a_t that could change from state s_t to s_{t+1}. $\sum_{a_{t+1}\in\mathcal{A}} F_{t+1}'$ indicates the outflow of s_{t+1}, i.e., the sum of flows passing through it with all possible actions $a_{t+1} \in \mathcal{A}$.

For simplicity, we summarize our GFlowGNN algorithm in Algorithm 1. Starting from an empty label set $\mathcal{V}_{\text{label}} = \emptyset$, for every iteration, GFlowGNN samples a valid action based on the generative flow network, and labels a new node to make a state transition $s_t \to s_{t+1}$. Then we update $\mathcal{V}_{\text{label}}^{t+1} = \mathcal{V}_{\text{label}}^t \cup \{v_{t+1}\}$ and update the classification GNN f one step accordingly. After that, we calculate the dynamic node vector and flow feature to update $\mathbf{v}(s_{t+1})$ and $\mathbf{\Phi}(s_{t+1})$. This process continues until the budget b is used up. After that we train GNN f with $\mathcal{V}_{\text{label}}^b$ until convergence and calculate $r(V_{\text{label}}^b)$ using (12). The policy network is then updated based on flow matching loss using (15).

6 Experiment

In this section, we present experimental results in terms of performance, exploration capability, and transferability to demonstrate the advantages of the proposed GFlowGNN. We use Cora, Citeseer, Pubmed and 5 Reddit (large social networks) as datasets to verify the effectiveness of the proposed algorithm, more details about the datasets can be found in the supplementary material.

Algorithm 1 GFlowGNN Algorithm

```
Require: : $\mathcal{V}_{\text{train}}$ training dataset; $f$: GNN; $b$: Budget; $B$: batch size;
    $E$: Epoch number; $\eta$: learning rate
1:  Initialize $\mathcal{V}_{\text{label}}^0 = \emptyset$
2:  repeat
3:      repeat {parallel do with a batch size $B$}
4:          Sample a valid action $a_t$: $v_{t+1} \sim \pi(a_t|s_t)$ s.t. $v_{t+1} \in \mathcal{V}_{\text{train}} \backslash \mathcal{V}_{\text{label}}^t$
5:          Make a state transition $s_{t+1} = T(s_t, a_t)$
6:          Update the labeled node set $\mathcal{V}_{\text{label}}^{t+1} = \mathcal{V}_{\text{label}}^t \cup \{v_{t+1}\}$
7:          Update GNN $f$ with $\mathcal{V}_{\text{label}}^{t+1}$ one step and calculate the node
            vector $\mathbf{v}(s_{t+1})$ and flow feature $\mathbf{\Phi}(s_{t+1})$
8:          Find the parent set $\mathcal{S}_p(s_{t+1})$ of $s_{t+1}$ and calculate their
            state matrices
9:      until $|\mathcal{V}_{\text{label}}^t| = b$
10:     Train GNN $f$ with $\mathcal{V}_{\text{label}}^b$ until convergence
11:     Calculate $r(\mathcal{V}_{\text{label}}^b)$ according to (12)
12:     Update the network parameter $\theta$ based on $\nabla\mathcal{L}_\theta(\tau)$ and $\eta$
        according to (15)
13: until epoch number $E$ is reached
Ensure: Policy $\pi(a_t|s_t)$ and the best labeled node set
```

6.1 Baselines

We use *Random*, *Uncertainty-based policy*, *Centrality-based policy*, *Coreset [Veličković* et al.*, 2017]*, *AGE [**?**]*, *ANRMAB [Gao* et al.*, 2018a]*, *IGP [Zhang* et al.*, 2022b]* as baselines for comparisons.

6.2 Evaluation Metrics and Parameters

We evaluate the performance of each method using common evaluation criteria (Micro-F$_1$, Macro-F$_1$ and accuracy). Following [Hu *et al.*, 2020a], we set the validation size and test size to 500 and 1000, respectively. In the testing phase, we run the results 1000 times and record the average evaluation result. For a single active graph learning task, we use MLP (4 layers, including one input layer, two hidden layers of dimension 128, and one output layer) as the structure of our neural network. For the transfer graph learning task, we adopt a combination of two-layer GCN and linear layers. The dimension of each layer in GCN embedding is 8, and the linear layer is the mapping from the GCN embedding dimension to 1. We set the budget to $5C$ (C is the number of classes corresponding to the dataset). We use Adam as the optimizer with a learning rate of 1e-3.

6.3 Performance Comparison

Active learning on different datasets with the same labeling budget: We first demonstrate the performance advantages of GFlowGNN on commonly used datasets. In this experiment, the training and test datasets are the same. There are two different methods to select label sets: 1) For the learning based methods, after training the model, we initialize 1000 different classification GNNs with different node sets and then use the trained policy to select label sets; 2) For the non-learning based methods, we directly apply their policy to select label sets. After that, we train 1000 classification GNNs based on these selected label sets and obtain the prediction performance on the test datasets. We show the average accuracy of each algorithm in Table 1. We can see that our approach could attain the highest accuracy among all datasets.

Method	Metric	Pubmed	Cora	Citeseer
Random	acc (%)	68.35	67.66	60.44
Uncertainty	acc (%)	69.52	60.36	60.34
Centrality	acc (%)	67.91	70.66	60.71
Coreset	acc (%)	65.26	64.15	45.81
IGP	acc (%)	79.5	77.9	69.5
AGE	acc (%)	74.78	71.53	66.61
ANRMAB	acc (%)	69.35	69.19	62.67
GPA	acc (%)	77.80	75.43	67.03
Ours	acc (%)	**81.04**	**78.67**	**70.58**

Table 1: The text accuracy on different datasets with the same labeling budget.

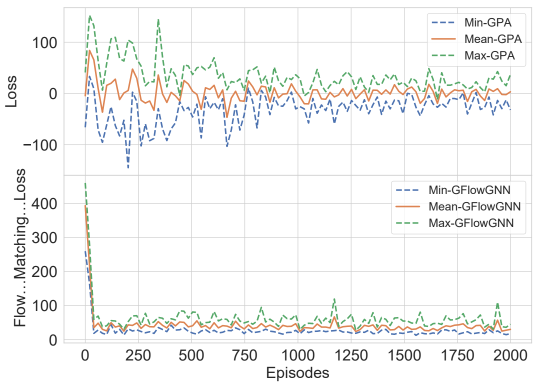

Figure 2: Losses of GFlowGNN and GPA on Citeseer

Figure 2 shows the training losses of GFlowGNN and GPA on Citeseer dataset. We can see that the loss of GFlowGNN drops significantly faster than that of RL-based method, highlighting the performance advantage of GFlowGNN in learning speed.

6.4 Exploration Capability Comparison

Figure 3 shows the number of high-quality label sets generated by GFlowGNN and GPA after training. We define a high-quality label set as the classification graph network whose node classification prediction accuracy exceeds a threshold after the label set is fed to the classification graph network and trained to converge. We define different thresholds for different datasets. For cora, citeseer, and pubmed, the thresholds are set to 0.8, 0.7, and 0.8, respectively. It can be clearly seen from the figure that on different datasets (cora corresponds to solid line, citeseer corresponds to dotted line, pubmed corresponds to dashed line), GFlowGNN's ability to generate high-quality label sets far exceeds GPA under the same number of generation, which demonstrate that GFlowGNN has better exploration ability.

Figure 4 shows the accuracy of the classification graph network corresponding to the optimal label set generated by GFlowGNN and GPA as the number of generated label sets grows. Clearly, Under the same number of explorations, GFlowGNN is able to generate label sets that make classifica-

Method	Metric	Pubmed	Reddit1	Reddit2	Reddit3	Reddit4	Reddit5
Random	Micro-F$_1$	68.35	81.88	91.19	87.76	85.37	86.45
	Macro-F$_1$	67.57	80.26	89.92	86.12	80.89	84.52
AGE	Micro-F$_1$	74.78	83.76	92.56	90.61	86.94	87.73
	Macro-F$_1$	73.26	82.81	91.61	89.99	83.15	85.88
ANRMAB	Micro-F$_1$	69.35	81.25	88.74	85.26	83.14	83.65
	Macro-F$_1$	68.68	79.43	86.58	83.06	76.8	79.99
GPA	Micro-F$_1$	77.80	88.10	95.19	92.07	91.39	90.66
	Macro-F$_1$	75.66	87.75	95.00	91.77	89.60	90.22
Ours	Micro-F$_1$	**77.89**	**89.42**	**95.75**	**92.97**	**92.13**	**91.14**
	Macro-F$_1$	**76.95**	**89.25**	**95.56**	**92.71**	**90.59**	**90.76**

Table 2: Transferable active learning results for different domain graphs

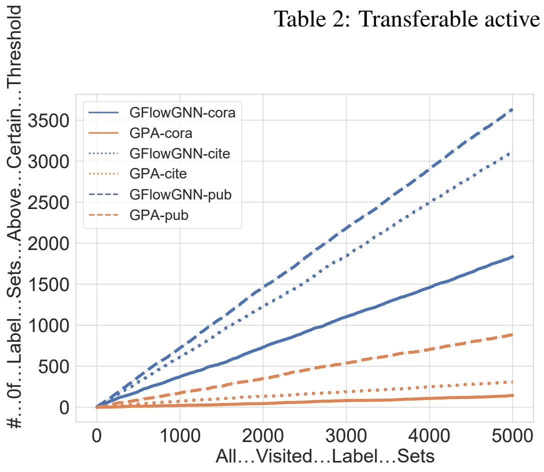

Figure 3: Number of high-quality label sets generated by GFlowGNN and GPA.

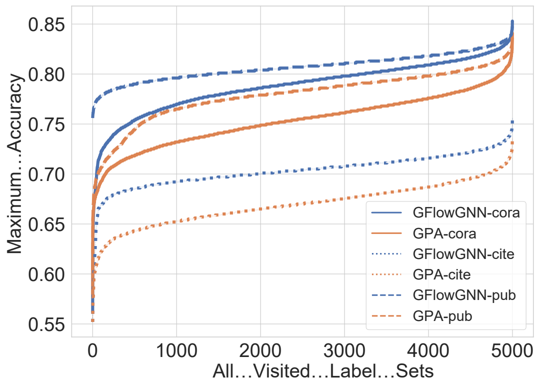

Figure 4: Maximum accuracy of label sets generated by GFlowGNN and GPA.

tion graph networks more accurate, proving that our method not only outperforms GPA on average, but also generates ultra-high-quality label sets that cannot be generated by existing methods.

6.5 Transferability Comparison

Transferable active learning on graphs from the different domains: Table 2 presents the performance on the different domains transfer graph learning task. We drop three heuristics and non-learning-based IGP methods as they are less relevant to the transfer task. All learning methods are trained on Cora and Citeseer, and tested on Pubmed and Reddit1$\sim$Reddit5. Experimental results show that our method can obtain a transferable policy with better performance than others. Compared with the both Micro-F$_1$ and Macro-F$_1$ of other algorithms, GFlowGNN always performs the best and GPA is slightly worse. Because the cross-domain transfer task is more difficult than the same-domain transfer task, the advantages of our method are more obvious in this task, which again convinces the strong transferability of our approach. Rather than maximizing cumulative reward, we focus more on the consistency of action probabilities with the true reward distribution.

7 Conclusion

In this work, by taking advantage of the strong generation ability of GFlowNets, we proposed a novel approach, named GFlowGNN, to model the graph active learning problem as a generation problem. We proposed the concepts of flow nodes and flow features to balance efficiency and accuracy. GFlowGNN learns the best policy to select the valuable node set, considered a compositional object, by sequential actions. We conduct extensive experiments on real datasets to convince the superiority of GFlowGNN on performance, exploration capability, and transferability.

Limitations: For some GNN scenarios where reward maximization is required, the performance of GFlowNets may be not as good as that of RL, because RL is oriented towards reward maximization rather than proportional to reward. Therefore, in GFlowGNN, the reward needs to be specially designed. Nevertheless, experiments show that the reward we designed has good performance and good generalization performance. GFlowGNN is the first work to apply GFlowNets into the GNN-related task, we believe our solution could prompt further developments in this area.

References

[Bengio *et al.*, 2021a] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

[Bengio *et al.*, 2021b] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *arXiv preprint arXiv:2111.09266*, 2021.

[Burbidge *et al.*, 2007] Robert Burbidge, Jem J Rowland, and Ross D King. Active learning for regression based on query by committee. In *International conference on intelligent data engineering and automated learning*, pages 209–218. Springer, 2007.

[Cai *et al.*, 2017] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085*, 2017.

[Chen *et al.*, 2019] Xia Chen, Guoxian Yu, Jun Wang, Carlotta Domeniconi, Zhao Li, and Xiangliang Zhang. Activehne: Active heterogeneous network embedding. *arXiv preprint arXiv:1905.05659*, 2019.

[Du *et al.*, 2015] Bo Du, Zengmao Wang, Lefei Zhang, Liangpei Zhang, Wei Liu, Jialie Shen, and Dacheng Tao. Exploring representativeness and informativeness for active learning. *IEEE transactions on cybernetics*, 47(1):14–26, 2015.

[Gao *et al.*, 2018a] L. Gao, H. Yang, C. Zhou, J. Wu, and Y. Hu. Active discriminative network representation learning. In *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18*, 2018.

[Gao *et al.*, 2018b] Li Gao, Hong Yang, Chuan Zhou, Jia Wu, Shirui Pan, and Yue Hu. Active discriminative network representation learning. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[Garg *et al.*, 2020] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

[Gu *et al.*, 2013] Quanquan Gu, Charu Aggarwal, Jialu Liu, and Jiawei Han. Selective sampling on graphs for classification. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 131–139, 2013.

[Hao *et al.*, 2020] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. Asgn: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 731–752, 2020.

[Hu *et al.*, 2020a] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. Graph policy network for transferable active learning on graphs. *Advances in Neural Information Processing Systems*, 33:10174–10185, 2020.

[Hu *et al.*, 2020b] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867, 2020.

[Jain *et al.*, 2022] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[Ji and Han, 2012] Ming Ji and Jiawei Han. A variance minimization criterion to active learning on graphs. In *Artificial Intelligence and Statistics*, pages 556–564. PMLR, 2012.

[Jiang and Gupta, 2021] Heinrich Jiang and Maya R Gupta. Bootstrapping for batch active sampling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3086–3096, 2021.

[Jin *et al.*, 2020] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Lahlou *et al.*, 2023] Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-García, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. *arXiv preprint arXiv:2301.12594*, 2023.

[Li *et al.*, 2020] Yayong Li, Jie Yin, and Ling Chen. Seal: Semisupervised adversarial active learning on attributed graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7):3136–3147, 2020.

[Li *et al.*, 2022] Wenqian Li, Yinchuan Li, Shengyu Zhu, Yunfeng Shao, Jianye Hao, and Yan Pang. Gflowcausal: Generative flow networks for causal discovery. *arXiv preprint arXiv:2210.08185*, 2022.

[Li *et al.*, 2023a] Wenqian Li, Yinchuan Li, Zhigang Li, Jianye Hao, and Yan Pang. Dag matters! gflownets enhanced explainer for graph neural networks. *arXiv preprint arXiv:2303.02448*, 2023.

[Li *et al.*, 2023b] Yinchuan Li, Shuang Luo, Yunfeng Shao, and Jianye Hao. Gflownets with human feedback. *arXiv preprint arXiv:2305.07036*, 2023.

[Li *et al.*, 2023c] Yinchuan Li, Shuang Luo, Haozhi Wang, and Jianye Hao. Cflownets: Continuous control with generative flow networks. *arXiv preprint arXiv:2303.02430*, 2023.

[Liu *et al.*, 2019] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[Liu *et al.*, 2020] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 338–348, 2020.

[Liu *et al.*, 2021] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *International Conference on Machine Learning*, pages 6837–6849. PMLR, 2021.

[Ma *et al.*, 2022] Handong Ma, Changsheng Li, Xinchu Shi, Ye Yuan, and Guoren Wang. Deep unsupervised active learning on learnable graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[Melville and Mooney, 2004] Prem Melville and Raymond J Mooney. Diverse ensembles for active learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 74, 2004.

[Nguyen and Smeulders, 2004] Hieu T Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79, 2004.

[Nishikawa-Toomey *et al.*, 2022] Mizu Nishikawa-Toomey, Tristan Deleu, Jithendaraa Subramanian, Yoshua Bengio, and Laurent Charlin. Bayesian learning of causal structure and mechanisms with gflownets and variational bayes. *arXiv preprint arXiv:2211.02763*, 2022.

[Sarlin *et al.*, 2020] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[Tang *et al.*, 2002] Min Tang, Xiaoqiang Luo, and Salim Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 120–127, 2002.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wu *et al.*, 2020a] Jian Wu, Victor S Sheng, Jing Zhang, Hua Li, Tetiana Dadakova, Christine Leon Swisher, Zhiming Cui, and Pengpeng Zhao. Multi-label active learning algorithms for image classification: Overview and future promise. *ACM Computing Surveys (CSUR)*, 53(2):1–35, 2020.

[Wu *et al.*, 2020b] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[Yang *et al.*, 2015] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127, 2015.

[Ying *et al.*, 2019] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[You *et al.*, 2019] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pages 7134–7143. PMLR, 2019.

[You *et al.*, 2020] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

[Zhang *et al.*, 2019] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[Zhang *et al.*, 2020] Wentao Zhang, Jiawei Jiang, Yingxia Shao, and Bin Cui. Snapshot boosting: a fast ensemble framework for deep neural networks. *Science China Information Sciences*, 63(1):1–12, 2020.

[Zhang *et al.*, 2021] Wentao Zhang, Yexin Wang, Zhenbang You, Meng Cao, Ping Huang, Jiulong Shan, Zhi Yang, and Bin Cui. Rim: Reliable influence-based active learning on graphs. *Advances in Neural Information Processing Systems*, 34:27978–27990, 2021.

[Zhang *et al.*, 2022a] Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pages 26412–26428. PMLR, 2022.

[Zhang *et al.*, 2022b] Wentao Zhang, Yexin Wang, Zhenbang You, Meng Cao, Ping Huang, Jiulong Shan, Zhi Yang, and Bin Cui. Information gain propagation: a new way to graph active learning with soft labels. *arXiv preprint arXiv:2203.01093*, 2022.

[Zhou *et al.*, 2020] 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.

[Zhu *et al.*, 2008] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1137–1144, 2008.

[Zhu *et al.*, 2020] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.