Tackling Provably Hard Representative Selection via Graph Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Representative Selection (RS) is the problem of finding a small subset of exemplars from a dataset that is representative of the dataset. In this paper, we study RS for unlabeled datasets and focus on finding representatives that optimize the accuracy of a model trained on the selected representatives. Theoretically, we establish a new hardness result for RS by proving that a particular, highly practical variant of it (RS for Learning) is hard to approximate in polynomial time within any reasonable factor, which implies a significant potential gap between the optimum solution of widely-used surrogate functions and the actual accuracy of the model. We then study a setting where additional information in the form of a (homophilous) graph structure is available, or can be constructed, between the data points. We show that with an appropriate modeling approach, the presence of such a structure can turn a hard RS (for learning) problem into one that can be effectively solved. To this end, we develop *RS-GNN*, a representation learning-based **RS** model based on **G**raph **N**eural **N**etworks. Empirically, we demonstrate the effectiveness of RS-GNN on problems with predefined graph structures as well as problems with graphs induced from node feature similarities, by showing that RS-GNN achieves significant improvements over established baselines on a suite of eight benchmarks.

1 Introduction

In the age of massive data, having access to tools that can select exemplar data points representative of an entire dataset is of crucial importance. *Representative selection* (RS) [35], finding a small subset of exemplars from an unlabeled dataset that transmits maximal information for a certain objective, has numerous applications in summarization, active learning, data compression, model training cost reduction, and many other domains (see, e.g., [8, 63, 78, 106, 114, 24, 66, 34]).

We first study the computational complexity of a specific but widely-applicable formulation of the RS problem, where we attempt to find a fixed-size subset of representative exemplars from a dataset that can be used to train a model with the best possible *accuracy* on the entire dataset. We show it is impossible to provide a polynomial-time RS algorithm with an approximation factor better than $\omega(n^{-1/\mathrm{poly}\log\log n})$, unless the Exponential Time Hypothesis (ETH) fails. ETH is a widely-believed assumption in the domain of parameterized complexity which states that the 3-SAT problem cannot be solved in subexponential time in the worst case. Note that $\omega(n^{-1/\mathrm{poly}\log\log n})$ is almost polynomial, ruling out the existence of any constant approximation or even poly-logarithmic approximation.

Our subconstant hardness result is of particular importance because several previous works find representatives by optimizing *surrogate functions*—these can be approximated well in theory—instead of the actual model accuracy. For instance, they consider a submodular surrogate function which can be approximated within a factor $1 - 1/e$ in polynomial time [47, 105, 83, 23, 31]. Our hardness result implies that, in the worst case, there is a significant gap between the optimum solution

Submitted to ICML 2023 Workshop: Sampling and Optimization in Discrete Space. Do not distribute.

of such surrogate functions and the actual accuracy of the model, rendering the surrogate functions poor estimators for the quality of the model. To the best of our knowledge, this is the first subconstant hardness result for the RS problem. This motivates us to deviate from directly defining proxy functions, and make use of learning-based approaches that discover the hidden structure of the data to guide the selection. This is in line with the recent attempts to solve computationally hard problems with neural networks, e.g., [107, 27].

We therefore study a setting where besides having access to data point features, we also have access to additional information about the data points that can help guide the selection process. Specifically, in this paper we assume the extra information is in the form of a (homophilous) graph structure between the data points. We show empirically that with an appropriate modeling approach, the presence of such a graph structure can turn an originally hard RS problem into one that can be effectively solved. To this end, we develop RS-GNN: a learning-based model for **R**epresentatives **S**election via **G**raph **N**eural **N**etworks. We first demonstrate the effectiveness of RS-GNN for selecting representative nodes from datasets where a natural graph can be accessed, i.e., where edges may be specified by some natural property of the data (e.g., paper citations). Then, we demonstrate that even when a natural graph is not available, creating a similarity graph of the input data points and applying RS-GNN can still select high-quality representatives. We conduct experiments on eight datasets with different sizes and properties, and in both settings where we have and do not have access to a graph structure. Our results show that our model provides significant improvements over three kinds of baselines: 1) well-established baselines that optimize predefined surrogate functions, 2) learning-based methods utilizing graph clustering/pooling and 3) baselines based on active learning.

Our main contributions are: 1) Providing a hardness result establishing that, under a standard computational-complexity assumption, RS is hard to approximate in polynomial time within any reasonable factor (this is the *first* subconstant hardness result for RS, to the best of our knowledge), 2) Demonstrating empirically that the existence of additional information in the form of a graph structure can make hard RS problems effectively solvable, and 3) Developing RS-GNN for effective RS when one has access to such a graph structure and showing its merit for datasets with natural and/or similarity graphs.

2 Related Work

We group the existing work that relates to our paper as follows (see Appendix B for more).

Active learning: In active learning [95, 26] we have an unlabeled set of data points that we can request to label. Since labeling is an expensive task, we usually have a limited budget, say, we can label up to k data points, which are then used to predict the labels of all data points. The goal is to select the set of data points to label in such a way as to maximize the accuracy of the final model. The data points can be iteratively selected in mini-batches (select a mini-batch, label the data points in the batch, update the model, and repeat), or in one-shot [54, 48, 19, 25, 5]. The latter is typically used when model training is time-consuming. RS can be used in the context of one-shot active learning, or for selecting the first mini-batch in the context of mini-batch active learning. In these contexts, a common approach to active learning is to use unsupervised surrogate functions such as KMediod [93] and MaxCover [53] to select a set of data points that maximally cover the dataset with respect to some objective. Moreover, active learning models have been developed for attributed graphs both for mini-batched labeling [17, 41] and for one-shot [109, 118]. We compare against many surrogate functions as well as one-shot graph active learning models in our experiments.

Hardness of Clustering: The hardness problem studied in this work is distantly related to clustering, which has come in many flavors and shapes: flat vs. hierarchical, partitioning vs. overlapping, graph-based vs. embedding-based vs. time-series-based, supervised vs unsupervised, etc. The interested reader may refer to references (e.g., [50, 62, 39, 111, 58, 110, 32]) for further information. We do emphasize here, though, that the most similar clustering objectives to what we study here are the center-based clustering problems such as k-center and k-means. In these settings, the hardness results and known algorithmic guarantees (i.e., lower and upper bounds) are not far from each other. For example, while non-metric k-center cannot be approximated to within any constant, the metric special case (generalizing the ubiquitous Euclidean setting) admits a 2-approximation [45] and cannot be approximated to within better than a factor 2 [102]. On the other hand, the k-means objective admits a constant-factor approximation [4, 61] and the best hardness results are 1.0013 [7, 71]. In contrast

to all these results, we present a superconstant hardness for the RS problem, stressing the big gap between any optimizable surrogate function and the true objective.

Graph clustering (community detection): Early approaches only considered the graph structure and disregarded the node features. These approaches typically learn an embedding for each node (e.g., the spectral features, random walk embeddings, or auto-encoder based embeddings) and then feed these node embeddings into a clustering algorithm such as kmeans (see, e.g., [18, 84, 46, 112]). Recently, approaches based on GNNs, which take both graph structure and node features into account, have gained more popularity and success [119, 91, 76, 104, 12, 60, 100]. RS and clustering are two highly related tasks: many models developed for RS can (in theory) be used for clustering and vice versa. However, due to the distinct properties of the two tasks, a model that works well for one task may not necessarily work well for the other. While our main focus is on RS, we also experiment with graph clustering and compare against several existing approaches.

3 Notation & Problem Definition

We use bold lowercase letters to denote vectors and bold uppercase letters to denote matrices. Let $\boldsymbol{x}_i$ represent the i^{th} element of $\boldsymbol{x}$ and $\boldsymbol{M}_i$ represent the i^{th} row of $\boldsymbol{M}$. For a function $f : A \mapsto B$ and a subset $A' \subseteq A$, we use $f|_{A'}$ to denote the restriction of the domain of f to A'. For a dataset, we use $\mathcal{V} = \{v_1, \ldots, v_m\}$ to represent the set of data points (of size m) and $\boldsymbol{X}$ to represent the data matrix, such that $\boldsymbol{X}_i$ corresponds to the features of v_i. When data points have class labels, we use c to represent the number of classes. First, we define a general framework for Representative Selection (RS) as follows:

Definition 3.1 (RS)**.** *Given a set of data points $\mathcal{V}$, their features $\boldsymbol{X}$, a number $0 < k \leq |\mathcal{V}|$, and a utility function $u : 2^{\mathcal{V}} \mapsto \mathbb{R}$, the representative selection problem is to select a subset $\mathcal{S} \subseteq \mathcal{V}$ of k representatives that maximize the utility $u(\mathcal{S})$.*[1]

The applicability and tractability of an RS problem depends on the utility function, u. Intuitively, u should capture the usefulness of the subset $\mathcal{S}$ as a representative of $\mathcal{V}$; more precisely, if there is a particular application of the full dataset $\mathcal{V}$, u quantifies the degree to which $\mathcal{S}$ can be used instead. This can vary with the particular application considered; in this paper we are mostly concerned with a particular utility model that associates representativeness with *learnability*. In the *Representative Selection for Learning* (RSL) problem, we get to see the labels of the selected representatives, based on which we aim to train a classifier with highest predictive performance on the entire dataset.

Definition 3.2 (RSL)**.** *Let $\mathcal{V}$ be a set of data points with observed features $\boldsymbol{X}$ and with labels Y generated from an oracle function $\phi^* : \mathcal{V} \mapsto \{1, \ldots, c\}$, and Φ be a class of predictor functions (not necessarily containing ϕ^*). Given $\mathcal{V}$, $\boldsymbol{X}$, Φ, y, and a number $0 < k \leq |\mathcal{V}|$, the goal of RSL is to select a subset $\mathcal{S}$ of k representatives from $\mathcal{V}$ such that training a classifier $\phi \in \Phi$ based on $\boldsymbol{X}$ and $Y|_{\mathcal{S}}$ maximizes the normalized accuracy of ϕ on the entire set $\mathcal{V}$.*

The model class Φ defines a suitable inductive hypothesis for the learning problem so that RS is well-defined. The predictor ϕ is chosen to maximize the normalized accuracy, defined as $\overline{\text{Acc}} = c(\text{accuracy} - 1/c)$, on the entire dataset; in other words it is a *transductive learning problem* [38]. RSL is a natural problem that has multiple real-world applications in dataset selection, active learning, efficient ML and other areas (see section 7 for examples). Further, it can be expected to correspond to a general notion of the representativeness of a subset, even outside a learning use case. In the rest of this paper we will focus on this version of the RS problem, and defer generalizations to other utility models to future work.

4 Theoretical Findings: Hardness Results for RSL

We describe a theoretical finding that explains why the common practice of hand designing and optimizing surrogate functions may not be a good approach for RSL. In Definition 3.2, let $u(S) = \overline{\text{Acc}}(\phi(S))$ represent the normalized accuracy of the classifier ϕ when trained on a subset $\mathcal{S}$ of data points. Let $\mathcal{S}^* = \arg\max_{\mathcal{S}} u(\mathcal{S})$ be the optimal set of representatives. Ideally, one would optimize $u(\mathcal{S})$ and find $\mathcal{S}^*$. This may, however, be impossible without a-priori having access to the labels for

[1] Note that $\mathcal{V}$ is implicit in the definition of u, hence in the definition of RS.

all data points. Alternatively, most existing works on RSL (and RS in general) focus on defining an intuitive surrogate function Ω and find a solution $\mathcal{S}^{\Omega}$ by optimizing $\Omega(\mathcal{S})$ in the hope that $u(\mathcal{S}^{\Omega})$ is a good approximator for $u(\mathcal{S}^*)$. In this section, we establish an inapproximability result demonstrating that $u(\mathcal{S}^{\Omega})$ may not be a good estimator of $u(\mathcal{S}^*)$ for any polynomial-time-computable surrogate function Ω. We do this by showing that there are naturally defined learning tasks for which there exists a significant gap between $u(\mathcal{S}^{\Omega})$ and $u(\mathcal{S}^*)$.

We start by studying the computational hardness of RSL on an end-to-end binary classification task, from which the aforementioned claims follow. We say an RS algorithm $\mathcal{A}$ (e.g., optimizing a surrogate function) approximates the optimal solution with an approximation factor α if we can establish that $u(\mathcal{S}^{\mathcal{A}})$ is within a multiplicative factor α of the optimal solution $u(\mathcal{S}^*)$ for any learning problem, where $\mathcal{S}^{\mathcal{A}}$ is the output of the RS algorithm.

Under the Exponential-Time Hypothesis (ETH) assumption[2], we show that there is no polynomial-time RS algorithm with an approximation factor better than $\omega(n^{-1/\text{poly}\log\log n})$. In other words, we show that there is an instance of RSL for which the gap between $u(\mathcal{S}^{\mathcal{A}})$ and $u(\mathcal{S}^*)$ for any polynomial-time RSL algorithm $\mathcal{A}$ is at least $\omega(n^{-1/\text{poly}\log\log n})$, unless ETH fails. A similar approximation gap exists between the best polynomial-time and best exponential-time algorithm. Note that $\omega(n^{-1/\text{poly}\log\log n})$ is almost polynomial, ruling out the existence of any constant approximation or even poly-logarithmic approximation. Also note that the best solution may not give 100% accuracy, nor does it necessarily match the accuracy obtained by using all labels (since we only use k).

Definition 4.1 (Fit-or-Not (FoN) Learning Problem)**.** *We have m data points and n binary features. Each feature is associated with one of two types: red or blue. The types are generated independently and uniformly at random, they are consistent across data points, and are hidden from the algorithm. Each data point has value 1 for two features and 0 for the rest. The label of a data point is 1 if the type of its features of value 1 are the same, and the label is 0 otherwise. The goal is to maximize the normalized accuracy for all the data points given labels only on selected data points.*

Figure 1 visually presents an instance of the FoN problem. Each row represents a data point and each column represents a feature. There are $m = 10$ data points, each having $n = 5$ binary features. Each data point has a value of 1 exactly for two of the features, and a value of 0 for the rest. The types of the first and the third features are *blue* and the types of the other three features are *red*. These types are hidden from the algorithm. For the first data point, the two values of 1 are for the first and second features. Since these features have different types, the label for this data point is 0. For the second data point, however, the two values of 1 are for the first and the third features that have the same type, therefore the label for this data point is 1. The labels of the other data points are determined similarly.

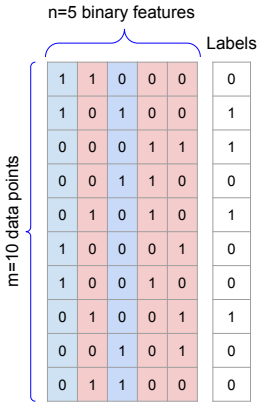


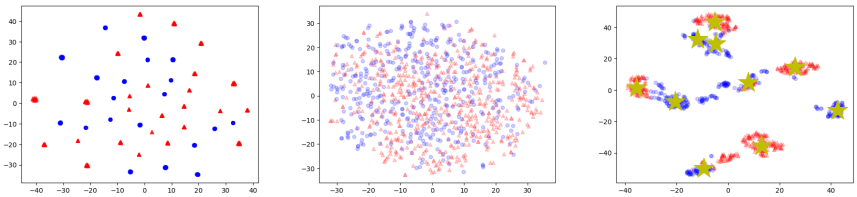
Figure 1: An instance of the FoN problem with $m = 10$ data points and $n = 5$ binary features.

The goal of the algorithm is to select a subset with $k < m$ data points in such a way that a model trained on the labels of those k data points makes accurate predictions for all the m data points (i.e. it generalizes well to the other $(m - k)$ data points).

FoN can be naturally framed as RSL by defining the components from Definition 3.2 as follows: let $\mathcal{V}$ be the set of m data points, $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ be the features matrix, Φ be the class of models that correspond to the data generation process in Definition 4.1 (i.e. constructed from particular partitions of features into red/blue etc.), k be the budget for selecting representatives, and let the labels ϕ^* be as defined above; the latent information consists of the types of the n features. The simplicity of FoN shows that RS is hard in a very broad form. The next theorem is the main result of this section.

Theorem 4.2. *There is no polynomial-time RSL algorithm for FoN with an approximation factor better than $\omega(n^{-1/\text{poly}\log\log n})$, unless the exponential-time hypothesis fails.*

[2] A widely-believed assumption (e.g., see [2, 15, 16, 22, 21, 28, 56, 59, 79, 80]) in the domain of parameterized complexity which states that 3-SAT cannot be solved in subexponential time in the worst case.



(a) (b) (c)

Figure 2: A t-SNE visualization of an instance of the FoN problem (the colors show the classes) when using (a) the original features, (b) adding a graph context and applying graph convolution, and (c) adding a graph context and applying a variant of RS-GNN to embed the nodes and select representatives (the yellow stars represent the selected representatives).

To the best of our knowledge, this is the first subconstant hardness result for any RS problem. This is of particular importance because several previous works have chosen to optimize *surrogate functions*—these can be approximated well in theory—instead of the actual model accuracy. For instance, previous work considers a submodular surrogate function which can be approximated within a factor $1 - 1/e$ in polynomial time [47, 105, 83, 23, 31]. Our hardness result implies the following.

Corollary 4.3. *In the worst case, there is a significant gap of* $\omega(n^{-1/\text{poly}\log\log n})$ *between* $u(\mathcal{S}^*)$ *and the solution of any polynomial-time approximable surrogate function that estimates* $u(\mathcal{S}^*)$.

The corollary follows because such surrogate functions can be optimized or approximated in polynomial time, but Theorem 4.2 shows that even for simple learning problems, approximating the accuracy is not possible in polynomial time (assuming the ETH); therefore, there are certain instances of the problem where optimizing for the surrogate functions does not optimize for the learning accuracy within the given approximation factor.

It is worth noting that many ML tasks are (known to be) hard to optimize, hence surrogate loss functions are commonly used in the context of deep learning: For example, finding a linear classifier with minimum 0-1 loss is NP-complete [81], and convex relaxations of this loss are typically used as surrogates in practice. What we show in this section is that fairly simple and natural instances of the representation selection problem are not only NP-complete but also hard to approximate. Thus the optimal solution of any surrogate function will be far from the actual optimum. This gap is prominent especially when we disentangle the task of finding the set to label from the task of training a model. We will see in the next sections that combining the two remedies the problem and produces very good results, which is in line with the applicability of surrogate loss functions within deep learning models.

5 RSL in Presence of a Graph Structure

In the previous section, we established the hardness of RSL by finding a natural problem called FoN for which RSL is hard to approximate in polynomial time within any reasonable factor. The hardness of RSL motivates seeking additional information about the problem that may help better guide the selection process. We next study whether the presence of additional information can help tackle the hard RSL problem effectively. In particular, we study the case where the additional information is provided in terms of graph structure of the data points with some degree of homophily; that is, nodes belonging to the same class are more likely to be connected to each other than nodes belonging to different classes. We present an algorithm for doing RSL in this setting and provide an empirical study and leave a theoretical analysis of this setting as well as settings with other types of additional information available as future work.

Let us start by providing a visual understanding of the FoN problem. We construct a version of the FoN problem with $m = 1000$ data points $n = 10$ features, where we assume the first 5 features have type red and the next 5 features have type blue. For each data point, we select two of its features uniformly at random and set their values to a number from $[0.9, 1.0]$ (other values are 0). In Figure 2(a), we present a visualization of the data points using t-SNE [101], where the colors represent the classes. From the visualization, we observe that there are 45 dense blocks of nodes (each having the same label) corresponding to $\binom{10}{2}$ different ways of selecting the position for the

non-zero elements. The blocks are scattered in such a way that it is difficult for an RSL algorithm to select a small subset such that a model trained on the labels of the nodes in that subset generalizes well to the other data points.

We now consider a variant of the FoN problem where an additional homophilous graph structure is available among the data points.

Definition 5.1 (GFoN)**.** *GFoN is a variant FoN where for each pair of data points we add an edge between them with probability p if they belong to the same class and p' if they belong to different classes. Letting $p > p'$ ensures the graph has some degree of homophily.*

A popular modeling choice to use when graph structure is available are graph convolution operations, where the features for each node are replaced by a weighted average of the features of their neighbors, with weights proportional to the degrees. If we apply a graph convolution operation on GFoN with $p = 0.05$ and $p' = 0.01$, we get the updated node features in Figure 2(b). One can observe that the nodes from the two classes have a high overlap and so any RSL algorithm may still fail. To understand why this happens, consider four data points whose non-zero values are in positions 1 and 2, 9 and 10, 1 and 9, and 2 and 10 respectively. The first two nodes will have a label of 1 and the other two nodes will have a label of 0. However, if we disregard the node degrees, aggregating the first two nodes will give the same representation as aggregating the second two nodes.

According to Figure 2(b), in presence of a graph structure, a naive application of graph convolution may not necessarily work best, and motivates the development of better modeling techniques. We show in our experiments that this also holds for many existing graph clustering/pooling approaches.

To better leverage the graph, in the next section we develop an approach that works by learning a mapping of the data points to a latent space where 1- we can group the nodes and select representatives that cover groups of nodes, and 2- we can better distinguish nodes belonging to each class. Figure 2(c) shows a t-SNE visualization of the data points in the latent space and the selected representatives. One can observe that with appropriate modeling, the presence of a homophilous graph structure turns the provably hard FoN problem into an RSL problem that can be effectively solved.

We next describe our approach for taking advantage of the graph context in improving RSL. In our experiments, we show that even when such a graph is not available, a similarity graph of the input node features may still be quite effective. Our theoretical and empirical results motivate obtaining or constructing graph contexts for RSL problems when possible.

6 RS-GNN: A Representation Learning-based Model for RSL

In this section we develop **RS-GNN**, a representation learning-based approach for **RS** via **GNN**s. We start with defining notation and provide necessary background. We denote an attributed graph as $\mathcal{G} = \{\mathcal{V}, \boldsymbol{A}, \boldsymbol{X}\}$ where $\mathcal{V} = \{v_1, \ldots, v_m\}$ represents the set of nodes, $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ represents the adjacency matrix, and $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ represents the matrix of node features (m nodes and n features).

Graph Neural Networks (GNNs) encode graph-structured data in continuous space [20]. Graph convolutional networks (GCNs) [67] are a powerful variant of GNNs. Let $\mathcal{G} = \{\mathcal{V}, \boldsymbol{A}, \boldsymbol{X}\}$ be an attributed graph, $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ be the adjacency matrix of $\mathcal{G}$ with self-loops included, and $\boldsymbol{D}$ be the degree matrix of $\hat{\boldsymbol{A}}$, where $\boldsymbol{D}_{ii} = \sum_j \hat{\boldsymbol{A}}_{ij}$ and $\boldsymbol{D}_{ij} = 0$ for $i \neq j$. The l^{th} layer of an L-layer GCN model with parameters $\boldsymbol{\Theta} = \{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L)}\}$ can be defined as: $\boldsymbol{H}^{(l)} = \sigma(\boldsymbol{D}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})$, where $\boldsymbol{H}^{(l)}$ represents node embeddings in the l^{th} layer ($\boldsymbol{H}^{(0)} = \boldsymbol{X}$), $\boldsymbol{W}^{(l)}$ is a weight matrix, and σ is an activation function. In the rest of the paper, we use $\mathsf{GCN}(\mathcal{G}; \boldsymbol{\Theta})$ to show the application of a GCN function with parameters $\boldsymbol{\Theta}$ on a graph $\mathcal{G}$.

Deep Graph Infomax (DGI) [103] is an approach for unsupervised representation-learning in attributed graphs. Given an attributed graph $\mathcal{G}$, in each iteration DGI creates a corrupted graph $\mathcal{G}'$ from $\mathcal{G}$. Then, it computes node embeddings $\boldsymbol{H}$ and $\boldsymbol{H}'$ for the two graphs by applying a GNN model on them, and a summary vector $\boldsymbol{s}$ based on the node embeddings $\boldsymbol{H}$. Finally, a discriminator is simultaneously trained to separate the node embeddings of the original graph (i.e., $\boldsymbol{H}$) from those of the corrupted graph (i.e., $\boldsymbol{H}'$) based on the summary vector $\boldsymbol{s}$. We describe the details of each step later when we define our final model.

6.1 Representative Selection via GNNs

An attributed graph $\mathcal{G} = \{\mathcal{V}, \boldsymbol{A}, \boldsymbol{X}\}$ has two modalities, the node features $\boldsymbol{X}$ and the graph structure $\boldsymbol{A}$. To be able to exploit both modalities in selecting good representatives, we employ a function EMB that combines the two modalities into a single embedding matrix of the graph. Concretely, $\mathsf{EMB}(\mathcal{G}) = \boldsymbol{H}$, where $\boldsymbol{H} \in \mathbb{R}^{m \times d}$ and d represents the embedding dimension. We also employ a differentiable function SEL that receives $\boldsymbol{H}$ as input and selects k nodes as representatives. That is, $\mathsf{SEL}(\boldsymbol{H}) = \mathcal{S}$. The two functions are optimized in a multi-task setting with the loss function $\mathcal{L} = \mathcal{L}_{\mathsf{EMB}} + \lambda \mathcal{L}_{\mathsf{SEL}}$, where $\mathcal{L}_{\mathsf{EMB}}$ encourages learning informative node embeddings and $\mathcal{L}_{\mathsf{SEL}}$ encourages selecting good representatives. One can create different RSL models with different choices of EMB, SEL, $\mathcal{L}_{\mathsf{EMB}}$ and $\mathcal{L}_{\mathsf{SEL}}$. GNNs have prove effective in learning node embeddings, so we use GNNs as our embedding function EMB. For $\mathcal{L}_{\mathsf{EMB}}$, we use the DGI objective which has shown to provide high-quality embeddings in unsupervised settings. For SEL, we consider a representative embedding matrix $\boldsymbol{R} \in \mathbb{R}^{k \times d}$ with learnable parameters where $\boldsymbol{R}$ is initialized randomly and $\boldsymbol{R}_j$ represents the embedding for the j^{th} representative. We let: $\mathcal{L}_{\mathsf{SEL}} = \sum_i \min_j(\mathsf{Dist}(\boldsymbol{H}_i, \boldsymbol{R}_j))$, where $\mathsf{Dist}(\boldsymbol{H}_i, \boldsymbol{R}_j)$ is the distance between the i^{th} node's embedding $\boldsymbol{H}_i$ and the j^{th} representative's embedding $\boldsymbol{R}_j$. We use Euclidean distance as the distance function. We select the representative corresponding to each $\boldsymbol{R}_j$ by finding the closest node embedding from $\boldsymbol{H}$ to $\boldsymbol{R}_j$, i.e., $\mathsf{argmin}_i(\mathsf{Dist}(\boldsymbol{H}_i, \boldsymbol{R}_j))$.

With the above loss function, the model can trivially reduce $\mathcal{L}_{\mathsf{SEL}}$ by making the values in $\boldsymbol{H}$ arbitrarily small. That is because multiplying $\boldsymbol{H}$ by a small constant may not change $\mathcal{L}_{\mathsf{EMB}}$ substantially, but it can make the distances between the nodes arbitrarily small, resulting in a low value for $\mathcal{L}_{\mathsf{SEL}}$ even for a random representative embedding matrix $\boldsymbol{R}$. We next describe a normalization scheme, *CenterNorm*, that is applied to $\boldsymbol{H}$ before $\boldsymbol{H}$ is used, which helps avoid this problem.

Algorithm 1 The training procedure of RS-GNN.

Input: $\mathcal{G} = (\mathcal{V}, \boldsymbol{A}, \boldsymbol{X})$, k

1: Initialize $\boldsymbol{R}$, $\boldsymbol{\Theta}$, and $\boldsymbol{U}$
2: **for** epoch=1 **to** #epochs **do**
3: $\quad \mathcal{G}' = (\mathcal{V}, \boldsymbol{A}, \mathsf{shuffle}(\boldsymbol{X}))$
4: $\quad \boldsymbol{H} = \mathsf{GCN}(\mathcal{G}; \boldsymbol{\Theta}),\ \ \boldsymbol{H}' = \mathsf{GCN}(\mathcal{G}'; \boldsymbol{\Theta})$
5: $\quad \boldsymbol{s} = \mathsf{sigmoid}(\frac{1}{n}\sum_i \boldsymbol{H}_i)$
6: $\quad \boldsymbol{p} = \mathsf{bilinear}(\boldsymbol{H}, \boldsymbol{s}; \boldsymbol{U}),\ \boldsymbol{p}' = \mathsf{bilinear}(\boldsymbol{H}', \boldsymbol{s}; \boldsymbol{U})$
7: $\quad \mathcal{L}_{\mathsf{EMB}} = -\sum_i (log(\boldsymbol{p}_i) + log(1 - \boldsymbol{p}'_i))$
8: $\quad \boldsymbol{\mu} = \frac{1}{n}\sum_i \boldsymbol{H}_i, \boldsymbol{\zeta} = \|\boldsymbol{H} - \boldsymbol{\mu}\|$
9: $\quad \tilde{\boldsymbol{H}} = \mathsf{CenterNorm}(\boldsymbol{H}) = (\boldsymbol{H} - \boldsymbol{\mu})/\boldsymbol{\zeta}$
10: $\quad \mathcal{L}_{\mathsf{SEL}} = \sum_i \min_j \mathsf{Dist}(\tilde{\boldsymbol{H}}_i, \boldsymbol{R}_j)$
11: $\quad \mathcal{L} = \mathcal{L}_{\mathsf{EMB}} + \lambda \mathcal{L}_{\mathsf{SEL}}$
12: $\quad$ Compute gradients for $\mathcal{L}$, upd. params.
13: Let $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{H}}$ be the representative and normalized node embeddings with minimum $\mathcal{L}$ during training.
14: **for** j=1 **to** k **do**
15: $\quad$ The j^{th} representative = $\mathsf{argmin}_i \mathsf{Dist}(\hat{\boldsymbol{H}}_i, \hat{\boldsymbol{R}}_j)$

CenterNorm: As we show in Appendix C, using the DGI loss function results in corrupted node embeddings that form a dense cluster in some part of the embedding space, and node embeddings (from the actual graph) that arrange themselves in subclusters around (and outside) this dense cluster of negative examples. Based on the this observation, we propose an ℓ_2 normalization of the node embeddings in $\boldsymbol{H}$ with respect to the center of the node embeddings:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_i \boldsymbol{H}_i,\ \boldsymbol{\zeta} = \|\boldsymbol{H} - \boldsymbol{\mu}\|,\ \tilde{\boldsymbol{H}} = (\boldsymbol{H} - \boldsymbol{\mu})/\boldsymbol{\zeta}, \tag{1}$$

where $\boldsymbol{\mu}$ is the center of the embeddings $\boldsymbol{H}$, $\boldsymbol{\zeta}$ is the ℓ_2 norms of the nodes with respect to the center, and $\tilde{\boldsymbol{H}}$ represents the normalized embeddings. With CenterNorm, the model can no longer decrease $\mathcal{L}_{\mathsf{SEL}}$ simply by making the values $\boldsymbol{H}$ smaller. Note that since the embedding clusters in $\boldsymbol{H}$ are at a large angle from each other (cf. supplementary material), ℓ_2 normalization has a low chance of collapsing two clusters. Furthermore, ℓ_2 normalization helps bring the nodes within one cluster closer to each other, which helps in identifying clusters and selecting representatives.

The Final RS-GNN Model: The full RS-GNN model is described in Algorithm 1. The input is an attributed graph $\mathcal{G} = (\mathcal{V}, \boldsymbol{A}, \boldsymbol{X})$ and a number k corresponding to the number of representative nodes that must be selected from the graph. The model initializes $\boldsymbol{R}$ (the representative embeddings), $\boldsymbol{\Theta}$ (the GCN parameters), and $\boldsymbol{U}$ (the parameters for the DGI discriminator). Lines 3 to 7 compute node embeddings $\boldsymbol{H}$ using a GCN [3] model and compute a DGI loss as $\mathcal{L}_{\mathsf{EMB}}$. Here, $\mathsf{bilinear}(\boldsymbol{H}, \boldsymbol{s}; \boldsymbol{U}) =$

[3] While we use GCN for direct comparability to existing work, note that one can use any other GNN model for RS-GNN. For example, for the FoN problem in Figure 2(c), we used a variant of the GraphSage [49] model as it better matched the problem.

$\mathsf{sigmoid}(\boldsymbol{H}^T\boldsymbol{U}\boldsymbol{s})$ where $\boldsymbol{H}^T$ indicates the transpose of $\boldsymbol{H}$. Lines 8 and 9 apply CenterNorm. Lines 10 to 12 compute $\mathcal{L}_{\mathsf{SEL}}$ and then $\mathcal{L}$ and update the parameters accordingly.

We keep track of the epoch with minimum joint loss $\mathcal{L}$. Let $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{H}}$ be the corresponding representative and normalized node embeddings in the best epoch. We select the j^{th} representative to be the node whose normalized embedding is closest to the j^{th} representative embedding. Lines 13 to 15 select the representatives based on $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{H}}$.

7 Empirical Results

We describe our baselines, datasets, and metrics, and defer implementation details to supplementary material[4].

Baselines: We compare against a representative set of baselines from different categories, as described below. The details for each baseline can be found in Appendix E. *Random:* Selects k nodes uniformly at random. *Popular:* Selects the k nodes with the maximum degree from the graph. *Surrogate functions on node features:* We test a representative set of surrogate functions: KMedoid, KMeans, Farthest First Search (FFS) [114] and (Greedy) MaxCover [53]. For KMeans, we select the closest node to each cluster center as a representative. For FFS and MaxCover, we select representatives sequentially. In FFS , the next representative is the node farthest away (by Euclidean distance) from the closest representative in the current set. In MaxCover , the next representative is the node that increases the coverage of the non-selected nodes the most. For MaxCover, we experiment with RBF kernel and cosine similarities; we use MC-RBF and MC-Cos to refer to the two versions respectively. Note that the sequential nature of FFS and MC makes them less amenable to parallelization. *Surrogate functions on node embeddings:* We use similar functions as above but apply them on DGI node embeddings as opposed to on the initial node features. Note that when we run these baselines using DGI embeddings as context, their selections are informed by both node features and the graph structure. *Graph clustering/pooling/active learning:* We compare against a number of graph clustering/pooling/active learning approaches from different categories. Specifically, we compare against MinCut [11] which is a well-established pooling approach, FeatProp [109] which is successful graph active learning approaches, SDCN [12] which is a well-established auto-encoder based graph clustering model, EGAE [36] and GCC [36] which are recent joint representation learning and clustering approaches, and DMoN [100] which is a state-of-the-art graph clustering approach based on modularity maximization.

Datasets: We use eight established benchmarks in the GNN literature: three citation networks namely Cora, CiteSeer, and Pubmed [94, 55], a citation network named OGBN-Arxiv [55] which is orders of magnitude larger than the previous three, two datasets from Amazon products (Photos and PC) [97], and two datasets from Microsoft Academic (CS and physics) [97]. Supplementary material offers a more detailed description of datasets and their statistics. Our datasets have a wide range in terms of the number of nodes (from 2K to 170K), edges (from 4.5K to 1.1M), features (from 100 to 8.5K) and classes (from 3 to 40).

Measures: We measure the quality of the selected representatives $\mathcal{S} \subseteq \mathcal{V}$ using the following transductive semi-supervised node-classification problem. We train a GCN model on the dataset where the parameters of the GCN are learned only based on the labels of the nodes in $\mathcal{S}$. Note that this GCN is completely independent of the internal GCN model used in RS-GNN. We randomly split the remaining nodes into validation and test sets. The validation set is used for early stopping. The classification accuracy on the test set is used as the metric for measuring the quality of the selected representatives. Considering a validation set for early stopping reduces the chances of overfitting for the classifier and makes the reported test accuracy mainly a function of the quality of the selected representatives.

RSL in the Presence of a Graph Structure: The results are presented in Table 1. For the results in this table, we set k for each dataset to be $2c$, where c represents the number of classes. We found this to be a small enough number for a meaningful comparison of the quality of the selected representatives [5], and high enough for the classification GCN model to learn appropriate functions of

[4]The code is available at: `https://github.com/google-research/google-research/rs_gnn`

[5]Note that if $k \approx n$, all models may perform equally well.

Table 1: For each dataset, each algorithm selects $2c$ representatives. Then, we train a GCN model on the labels of the selected representatives. The reported metric is the test accuracy of the GCN models. Bold numbers indicate statistically significant winner(s) following a t-test (p-value=0.05). The color-codes and symbols represent ✣ surrogate functions on features, ✤ surrogate functions on embeddings, and ❦ attributed graph clustering/pooling or active learning approaches.

	Selector	Cora	CiteSeer	Pubmed	Photos	PC	CS	Physics	Arxiv	Avg.
	Random	49.1±6.9	33.1±8.3	52.0±8.1	70.2±6.4	65.4±6.1	72.9±5.0	73.6±6.8	49.0±1.4	58.2
	Popular	59.2±1.3	35.5±0.8	63.3±0.3	34.9±0.5	49.9±2.1	73.1±1.1	50.5±0.0	31.3±0.8	49.7
✣	KMedoid	53.7±5.4	40.3±2.8	53.2±1.0	65.8±1.2	66.9±1.8	51.8±1.0	66.6±1.6	43.9±1.5	55.3
✣	KMeans	32.5±5.8	35.4±1.2	50.6±0.5	72.5±0.9	70.9±1.0	66.5±0.5	73.0±1.4	48.4±0.5	56.2
✣	FFS	48.5±8.0	39.3±6.8	43.1±4.9	80.0±5.0	71.5±3.4	54.6±2.2	76.6±2.8	45.2±0.8	57.3
✣	MC-RBF	45.5±2.7	25.8±3.7	53.0±0.2	78.4±1.2	65.5±0.9	66.4±1.2	58.1±0.3	51.4±0.6	55.5
✣	MC-Cos	49.7±9.5	50.2±3.1	66.6±0.5	77.2±1.0	**74.0**±3.7	87.3±1.4	81.3±3.4	47.6±1.6	70.0
✤	KMedoid	48.4±4.4	34.1±1.9	60.9±5.3	81.5±2.6	69.8±3.4	82.5±2.9	81.2±7.0	OOM	—
✤	KMeans	62.6±9.3	42.7±6.3	60.5±6.3	83.6±2.9	**74.8**±2.7	86.9±1.8	**90.6**±2.4	51.2±1.0	69.1
✤	FFS	62.6±4.5	50.4±5.7	46.7±7.2	73.4±5.7	63.5±6.4	84.8±5.3	83.4±5.0	48.6±2.2	64.2
✤	MC-RBF	66.3±2.6	35.3±4.5	54.9±5.3	37.4±3.7	50.7±2.5	65.2±1.2	59.8±5.1	41.2±1.6	51.4
✤	MC-Cos	67.3±5.2	49.0±4.1	**67.3**±1.1	84.4±1.0	**74.0**±3.7	87.3±1.4	81.3±3.4	47.6±1.6	70.0
❦	MinCUT	51.9±7.5	37.3±8.0	59.5±6.1	14.4±7.5	18.3±8.7	85.5±1.4	86.0±3.3	32.4±5.2	48.2
❦	FeatProp	56.6±1.7	37.8±1.6	65.2±0.6	78.2±1.8	68.5±1.1	74.7±0.3	81.4±0.6	47.8±1.0	63.8
❦	SDCN	41.6±9.5	33.8±9.3	47.8±8.5	61.0±10.8	54.2±8.7	66.4±6.7	77.5±9.2	38.4±4.4	52.6
❦	EGAE	64.4±3.8	45.0±5.8	57.7±5.1	83.5±3.0	**75.4**±3.2	79.9±3.2	80.4±4.0	50.6±1.0	67.1
❦	GCC	68.7±2.2	49.3±6.0	63.9±5.1	84.2±1.4	72.1±2.1	85.8±1.5	**89.6**±0.9	**52.1**±1.4	70.7
❦	DMoN	58.0±7.1	40.5±7.4	55.3±7.5	78.6±9.2	70.3±3.3	84.3±1.4	85.9±3.8	**52.5**±1.9	65.7
	RS-GNN	**72.4**±3.7	**54.7**±3.9	65.8±3.0	**86.3**±1.4	**74.3**±1.7	**89.3**±0.8	**90.0**±2.6	**52.6**±1.2	**73.2**

the data. Since c is different for each dataset, making k a function of c also provides the opportunity to compare performance not only in terms of variation in the datasets, but also in terms of variation in the number of selected representatives.

RS-GNN performs well across all datasets and consistently outperforms (or matches) the baselines (with the exception of Pubmed). It has a low variance across different runs making it a reliable model. Among the surrogate functions, MC-Cos and KMeans perform best. We found FFS to be sensitive to outliers. We also found it difficult to select a set of hyperparameters for MC-RBF that work well across datasets. Among graph clustering/pooling and active learning approaches, we found GCC to perform best and be the only model that (overall) outperforms surrogate functions when applied on DGI embeddings. Many of the other graph clustering/pooling or active learning approaches even fall short of the MC-Cos model when applied on the node features alone, thus showing the importance of developing appropriate models for taking advantage of the graph structure and confirming our finding in Figure 2. MinCut produced degenerate solutions for Photos and PC datasets in many runs (assigning all nodes to one cluster), hence performing poorly on them.

More results, analysis, and visualizations is provided in Appendix A.

8 Conclusion

In this paper, we studied the representative selection (RSL) problem theoretically and empirically. We proved new hardness results showing it is impossible to provide a polynomial-time algorithm for RSL with an approximation within any reasonable factor, unless the exponential time hypothesis fails. The hardness result explains the significant gap between the accuracy of models trained on optimal representatives and the widely-used surrogate functions for RS problems, and, in turn, justifies new techniques to solve this problem. In light of this result we proposed RS-GNN to optimize the RSL task via graph neural networks, and showed its effectiveness on a suite of different datasets and tasks.

References

[1] Zeinab Abbassi, Vahab S. Mirrokni, and Mayur Thakur. Diversity maximization under matroid constraints. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The*

19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013, pages 32–40. ACM, 2013.

[2] Divesh Aggarwal and Noah Stephens-Davidowitz. (gap/s) eth hardness of svp. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 228–238, 2018.

[3] Amir Hosein Khas Ahmadi. *Memory-based graph networks*. PhD thesis, University of Toronto (Canada), 2020.

[4] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *SIAM J. Comput.*, 49(4), 2020.

[5] Kareem Amin, Corinna Cortes, Giulia DeSalvo, and Afshin Rostamizadeh. Understanding the effects of batching in online active learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3482–3492. PMLR, 2020.

[6] Martin Aumüller, Sariel Har-Peled, Sepideh Mahabadi, Rasmus Pagh, and Francesco Silvestri. Fair near neighbor search via sampling. *SIGMOD Rec.*, 50(1):42–49, 2021.

[7] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. In *International Symposium on Computational Geometry*, 2015.

[8] Ramakrishna Bairi, Rishabh Iyer, Ganesh Ramakrishnan, and Jeff Bilmes. Summarization of multi-document topic hierarchies using submodular mixtures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 553–563, 2015.

[9] MohammadHossein Bateni, Hossein Esfandiari, and Vahab S. Mirrokni. Optimal distributed submodular optimization via sketching. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1138–1147. ACM, 2018.

[10] Aditya Bhaskara, Mehrdad Ghadiri, Vahab S. Mirrokni, and Ola Svensson. Linear relaxations for finding diverse elements in metric spaces. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4098–4106, 2016.

[11] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *ICML*, pages 874–883. PMLR, 2020.

[12] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *Proceedings of The Web Conference 2020*, pages 1400–1410, 2020.

[13] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. Recent advances and emerging challenges of feature selection in the context of big data. *Knowl. Based Syst.*, 86:33–45, 2015.

[14] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[15] Mark Braverman, Young Kun Ko, Aviad Rubinstein, and Omri Weinstein. Eth hardness for densest-k-subgraph with perfect completeness. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1326–1341. SIAM, 2017.

[16] Mark Braverman, Young Kun Ko, and Omri Weinstein. Approximating the best nash equilibrium in no (log n)-time breaks the exponential time hypothesis. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 970–982. SIAM, 2014.

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

[18] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900, 2015.

[19] Shayok Chakraborty, Vineeth Balasubramanian, and Sethuraman Panchanathan. Adaptive batch mode active learning. *IEEE transactions on neural networks and learning systems*, 26(8):1747–1760, 2014.

[20] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

[21] Lin Chen, Hossein Esfandiari, Gang Fu, Vahab S Mirrokni, and Qian Yu. Feature cross search via submodular optimization. *arXiv preprint arXiv:2107.02139*, 2021.

[22] Yijia Chen, Kord Eickmeyer, and Jörg Flum. The exponential time hypothesis and the parameterized clique problem. In *International Symposium on Parameterized and Exact Computation*, pages 13–24. Springer, 2012.

[23] Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. In *ICML*, pages 160–168. PMLR, 2013.

[24] Richard Church and Charles ReVelle. The maximal covering location problem. In *Papers of the regional science association*, volume 32, pages 101–118. Springer-Verlag, 1974.

[25] Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch active learning at scale. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[27] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

[28] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Lower bounds based on the exponential-time hypothesis. In *Parameterized Algorithms*, pages 467–521. Springer, 2015.

[29] Chris H. Q. Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. In *2nd IEEE Computer Society Bioinformatics Conference, CSB 2003, Stanford, CA, USA, August 11-14, 2003*, pages 523–529. IEEE Computer Society, 2003.

[30] S Durga, Rishabh Iyer, Ganesh Ramakrishnan, and Abir De. Training data subset selection for regression with controlled generalization error. In *ICML*, pages 9202–9212. PMLR, 2021.

[31] Hossein Esfandiari, Amin Karbasi, and Vahab Mirrokni. Adaptivity in adaptive submodularity. In *Conference on Learning Theory*, pages 1823–1846. PMLR, 2021.

[32] Absalom E. Ezugwu, Abiodun M. Ikotun, Olaide O. Oyelade, Laith Abualigah, Jeffery O. Agushaka, Christopher I. Eke, and Andronicus A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[33] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Dan Feldman. *Coresets and Their Applications*. Citeseer, 2010.

[35] Dan Feldman. Core-sets: Updated survey. In *Sampling techniques for supervised or unsupervised tasks*, pages 23–44. Springer, 2020.

[36] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. Efficient graph convolution for joint node representation learning and clustering. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 289–297, 2022.

[37] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

[38] A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 148–155, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[39] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, 2007.

[40] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[41] Li Gao, Hong Yang, Chuan Zhou, Jia Wu, Shirui Pan, and Yue Hu. Active discriminative network representation learning. In *IJCAI*, 2018.

[42] Xing Gao, Wenrui Dai, Chenglin Li, Hongkai Xiong, and Pascal Frossard. ipool–information-based pooling in hierarchical graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[43] Yunhao Ge, Yunkui Pang, Linwei Li, and Laurent Itti. Graph autoencoder for graph compression and representation learning. In *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*, 2021.

[44] Jonathan Godwin*, Thomas Keck*, Peter Battaglia, Victor Bapst, Thomas Kipf, Yujia Li, Kimberly Stachenfeld, Petar Veličković, and Alvaro Sanchez-Gonzalez. Jraph: A library for graph neural networks in jax., 2020.

[45] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[46] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[47] Andrew Guillory. *Active Learning and Submodular Functions*. University of Washington, 2012.

[48] Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. In *NIPS*, pages 593–600. Citeseer, 2007.

[49] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 30, 2017.

[50] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[52] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020.

[53] Dorit S Hochbaum and Anu Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.

[54] Steven CH Hoi, Rong Jin, Jianke Zhu, and Michael R Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on Machine learning*, pages 417–424, 2006.

[55] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[56] Joey Huchette, Haihao Lu, Hossein Esfandiari, and Vahab Mirrokni. Contextual reserve price optimization in auctions via mixed-integer programming. *arXiv preprint arXiv:2002.08841*, 2020.

[57] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. Composable core-sets for diversity and coverage maximization. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108. ACM, 2014.

[58] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall advanced reference series. Prentice Hall, 1988.

[59] Peter Jonsson, Victor Lagerkvist, Gustav Nordh, and Bruno Zanuttini. Complexity of sat problems, clone theory and the exponential time hypothesis. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1264–1277. SIAM, 2013.

[60] Barakeel Fanseu Kamhoua, Lin Zhang, Kaili Ma, James Cheng, Bo Li, and Bo Han. Grace: A general graph convolution framework for attributed graph clustering. *ACM Journal of the ACM (JACM)*, 2022.

[61] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, SCG '02, pages 10–18, New York, NY, USA, 2002. Association for Computing Machinery.

[62] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 2009.

[63] Vishal Kaushal, Rishabh Iyer, Suraj Kothawade, Rohan Mahadev, Khoshrav Doctor, and Ganesh Ramakrishnan. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1289–1299. IEEE, 2019.

[64] Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *ICML*, pages 5464–5474. PMLR, 2021.

[65] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *AAAI*, volume 35, pages 8110–8118, 2021.

[66] Krishnateja Killamsetty, Xujiang Zhao, Feng Chen, and Rishabh Iyer. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *NeurIPS*, 34, 2021.

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

[68] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NeurIPS*, pages 972–981, 2017.

[69] Ziyi Kou, Yang Zhang, Lanyu Shang, and Dong Wang. Faircrowd: Fair human face dataset sampling via batch-level crowdsourcing bias inference. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10, 2021.

[70] Dongha Lee, Su Kim, Seonghyeon Lee, Chanyoung Park, and Hwanjo Yu. Learnable structural semantic readout for graph classification. In *ICDM*, pages 1180–1185. IEEE, 2021.

[71] Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

[72] Jae-Gil Lee, Yuji Roh, Hwanjun Song, and Steven Euijong Whang. Machine learning robustness, fairness, and their convergence. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 4046–4047. ACM, 2021.

[73] J.G. Lee and C.G. Chung. An optimal representative set selection method. *Information and Software Technology*, 42(1):17–25, 2000.

[74] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. PMLR, 2019.

[75] Chuang Liu, Yibing Zhan, Chang Li, Bo Du, Jia Wu, Wenbin Hu, Tongliang Liu, and Dacheng Tao. Graph pooling for graph neural networks: Progress, challenges, and opportunities. *arXiv preprint arXiv:2204.07321*, 2022.

[76] Liang Liu, Zhao Kang, Jiajia Ruan, and Xixu He. Multilayer graph contrastive clustering network. *Information Sciences*, 613:256–267, 2022.

[77] Ning Liu, Songlei Jian, Dongsheng Li, Yiming Zhang, Zhiquan Lai, and Hongzuo Xu. Hierarchical adaptive pooling by capturing high-order dependency for graph representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[78] Yuzong Liu, Rishabh Iyer, Katrin Kirchhoff, and Jeff Bilmes. Svitchboard ii and fisver i: High-quality limited-complexity corpora of conversational english speech. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[79] Daniel Lokshtanov, Dániel Marx, Saket Saurabh, et al. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, (105):41–72, 2011.

[80] Pasin Manurangsi. Almost-polynomial ratio eth-hardness of approximating densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–961, 2017.

[81] Patrice Marcotte and Gilles Savard. Novel approaches to the discrimination problem. *ZOR — Methods and Models of Operations Research*, 36, 1992.

[82] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[83] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *ICML*, pages 6950–6960. PMLR, 2020.

[84] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[85] Jana Novovicová, Petr Somol, Michal Haindl, and Pavel Pudil. Conditional mutual information based feature selection for classification task. In Luis Rueda, Domingo Mery, and Josef Kittler, editors, *Progress in Pattern Recognition, Image Analysis and Applications, 12th Iberoamericann Congress on Pattern Recognition, CIARP 2007, Valparaiso, Chile, November 13-16, 2007, Proceedings*, volume 4756 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 2007.

[86] Feng Pan, Wei Wang, Anthony K. H. Tung, and Jiong Yang. Finding representative set from massive data. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 338–345. IEEE Computer Society, 2005.

[87] Yunsheng Pang, Yunxiang Zhao, and Dongsheng Li. Graph pooling via coarsened graph infomax. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2177–2181, 2021.

[88] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *NeurIPS*, 34:20596–20607, 2021.

[89] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830, 2011.

[90] Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014.

[91] Zhihao Peng, Hui Liu, Yuheng Jia, and Junhui Hou. Attention-driven graph clustering network. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 935–943, 2021.

[92] Yuji Roh, Kangwook Lee, Steven Euijong Whang, and Changho Suh. Fairbatch: Batch selection for model fairness. In *ICLR*, 2021.

[93] Erich Schubert and Peter J Rousseeuw. Faster k-medoids clustering: improving the pam, clara, and clarans algorithms. In *Similarity Search and Applications: 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2–4, 2019, Proceedings 12*, pages 171–187. Springer, 2019.

[94] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

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

[96] Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*, 2019.

[97] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[98] Shubhanshu Shekhar, Greg Fields, Mohammad Ghavamzadeh, and Tara Javidi. Adaptive sampling for minimax fair classification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *NeurIPS*, 2021.

[99] Tom AB Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of classification*, 1997.

[100] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *JMLR*, 2023.

[101] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008.

[102] V.V. Vazirani. *Approximation Algorithms*. Springer Berlin Heidelberg, 2013.

[103] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR*, 2(3):4, 2019.

[104] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: A deep attentional embedding approach. *arXiv preprint arXiv:1906.06532*, 2019.

[105] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *ICML*, pages 1954–1963. PMLR, 2015.

[106] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE, 2014.

[107] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

[108] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pages 6861–6871. PMLR, 2019.

[109] Yuexin Wu, Yichong Xu, Aarti Singh, Yiming Yang, and Artur Dubrawski. Active learning for graph neural networks via node feature propagation. *arXiv preprint arXiv:1910.07567*, 2019.

[110] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2, 2015.

[111] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[112] Fanghua Ye, Chuan Chen, and Zibin Zheng. Deep autoencoder-like nonnegative matrix factorization for community detection. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1393–1402, 2018.

[113] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *NeurIPS*, 31, 2018.

[114] Chong You, Chi Li, Daniel Robinson, and Rene Vidal. Self-representation based unsupervised exemplar selection in a union of subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[115] Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *ICLR*, 2020.

[116] Sepehr Abbasi Zadeh, Mehrdad Ghadiri, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Scalable feature selection via distributed diversity maximization. In Satinder P. Singh and Shaul Markovitch, editors, *AAAI*, pages 2876–2883, 2017.

[117] Liang Zhang, Xudong Wang, Hongsheng Li, Guangming Zhu, Peiyi Shen, Ping Li, Xiaoyuan Lu, Syed Afaq Ali Shah, and Mohammed Bennamoun. Structure-feature based graph self-adaptive pooling. In *Proceedings of The Web Conference*, pages 3098–3104, 2020.

[118] Wentao Zhang, Zhi Yang, Yexin Wang, Yu Shen, Yang Li, Liang Wang, and Bin Cui. Grain: Improving data efficiency of graph neural networks via diversified influence maximization. *arXiv preprint arXiv:2108.00219*, 2021.

[119] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. *arXiv preprint arXiv:1906.01210*, 2019.

A More results

A.1 Selecting More Representatives

We already presented results for the case where we select $2c$ representatives from each dataset, where c is the number of classes. Here, we present more results for the case where we select $5c$ representatives from each class. We limit our baselines to the ones that were either more competitive/informative or took less time to run. The results are in Table 2. We can observe that the performance for all models improves when more representatives are selected. We can also observe that the gap between the models shrinks when more representatives are selected (even the random baseline now shows a competitive performance). Nevertheless, RS-GNN shows a similar trend as when we selected $2c$ representatives and outperforms the baselines when averaged across datasets.

Table 2: For each dataset, each algorithm selects $5c$ representatives. Then, we train a GCN model on the labels of the selected representatives. The reported metric is the test accuracy of the GCN models. Surrogate function selectors mark in blue use DGI embeddings.

Selector	Cora	CiteSeer	Pubmed	Photos	PC	CS	Physics	Arxiv	Avg.
Random	66.3±4.6	47.3±6.6	62.0±8.3	84.0±3.5	76.8±3.4	83.9±2.3	84.4±5.8	54.6±1.2	69.9
Popular	64.1±0.6	43.2±1.4	63.4±0.9	43.9±1.1	54.1±1.4	78.6±0.5	50.5±0.0	42.6±1.4	55.0
KMedoid	62.2±1.3	42.1±3.1	60.8±0.3	78.8±0.6	74.4±1.2	63.9±1.7	64.3±0.6	50.6±0.5	62.1
KMeans	66.7±1.1	53.8±1.3	71.6±0.5	84.8±1.0	81.4±0.7	76.7±1.0	80.6±0.3	56.8±0.3	71.6
MC-Cos	71.3±2.8	59.0±2.8	64.8±0.3	87.5±0.4	78.3±0.7	88.4±0.2	92.4±0.4	56.2±0.5	74.7
KMeans	74.9±3.1	54.3±4.8	69.3±3.2	89.4±1.3	82.0±1.4	89.2±0.8	92.1±0.7	55.1±1.1	75.8
MC-Cos	75.8±1.7	59.8±2.5	70.6±2.4	90.0±1.2	82.3±1.1	89.4±0.8	86.1±1.9	55.8±0.7	76.2
MinCUT	66.8±4.3	48.7±6.6	69.0±4.2	16.7±8.1	16.5±10.2	88.6±1.2	92.3±1.0	55.3±7.5	56.7
DMoN	70.1±3.8	54.9±4.6	70.2±4.0	86.7±2.4	74.8±5.8	89.7±0.7	90.9±1.8	57.1±1.0	74.3
RS-GNN	77.3±1.9	62.7±2.3	68.7±2.4	90.6±0.5	83.0±1.6	90.1±0.6	92.4±0.8	56.6±1.2	77.7

A.2 Results: RSL in the Absence of a Graph Structure

In several applications, a natural graph may not be available. For semi-supervised node classification, it has recently been shown that even without a natural graph, one can still leverage GNNs by learning both a graph structure and GNN parameters simultaneously [33]. We extend the aforementioned results to the RSL problem. In particular, we assume we have access *only* to the node features of our datasets, and not to their graph structures. The baselines select representatives only based on the node features. For RS-GNN, we first create a kNN similarity graph between the nodes and then run the model on the node features and the created graph. The kNN graph is computed once and then fixed during training; we leave further experiments with learning the graph structure as future work. Notice that all models have access to the same context.

Once the representatives are selected, we train GCN classifiers on the labels of the selected nodes. For the baselines, we run our classification GCN in two settings: (1) the only edges in the graph are self-loops, (2) the edges in the graph are those from a kNN similarity graph. We call the former a multi-layer perceptron (MLP) and the latter a (kNN-)GCN. We report results for the top baselines that operate on node features *only*.

Since operating without graph structure reduces the signal in the datasets, we allow each model to select $5c$ representatives for the experiments in this section. The results are presented in Table 5. The results show that selecting representatives using RS-GNN performs consistently well on all datasets and provides a boost compared to our baselines on many of the datasets. This confirms that even a similarity graph can still be helpful in improving RSL and that RS-GNN is an effective RSL algorithm for datasets where a graph structure is not available.

A.3 Label Coverage

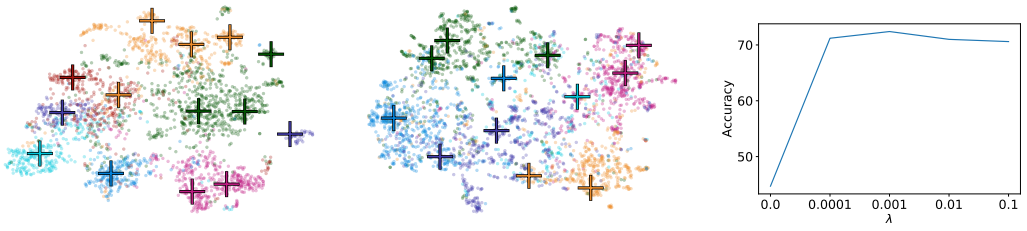
Once a set of representatives are selected from a dataset, we define a specific class label to be *covered* if at least one of the representatives belongs to that class. We define *label coverage* as the percentage of class labels that are covered by the selected representatives. In Table 3, we report the label coverage results for the baselines and our model when selecting $k = 2c$ representatives. RS-GNN performs well in terms of selecting points that cover all labels, with a coverage of 100.0% on four of the datasets. We observe an interesting phenomenon on the Arxiv dataset where (unlike other datasets) many baselines outperform RS-GNN in terms of label coverage (especially GCC), but RS-GNN shows a higher accuracy compared to the baselines. We believe this is due to the high label imbalance in this dataset (the largest class has 27321 examples and the smallest has 29 examples). Future work can extend our work for optimizing macro accuracy across classes.

A.4 Visualization

In Figure 3(a), we use UMAP [82] to visualize the learned embeddings and the selected representatives of RS-GNN for the Cora dataset. The colors in the plot represent the class to which the node or the

Table 3: Label coverage of different RS algorithms when $k = 2c$. The surrogate function selectors marked in blue use DGI embeddings.

Selector	Cora	CiteSeer	Pubmed	Photos	PC	CS	Physics	Arxiv
Random	86.4±10.8	84.2±12.7	85.0±17.0	80.6±11.1	64.0±9.4	72.7±9.4	75.0±14.3	55.0±4.9
Popular	85.7±0.0	50.0±0.0	66.7±0.0	37.5±0.0	30.0±0.0	66.7±0.0	20.0±0.0	15.0±0.0
KMedoid	100.0±0.0	100.0±0.0	66.7±0.0	87.5±0.0	80.0±0.0	73.3±0.0	60.0±0.0	50.0±0.0
KMeans	100.0±0.0	83.3±0.0	100.0±0.0	75.0±0.0	70.0±0.0	46.7±0.0	60.0±0.0	65.0±0.0
FFS	84.3±11.3	88.3±9.5	71.7±12.2	93.1±8.6	74.5±9.4	36.3±3.4	67.0±9.8	63.0±2.3
MC-RBF	100.0±0.0	100.0±0.0	66.7±0.0	75.0±0.0	60.0±0.0	66.7±0.0	60.0±0.0	52.5±0.0
MC-Cos	97.1±5.9	98.3±5.1	100.0±0.0	75.0±0.0	80.0±0.0	66.7±0.0	100.0±0.0	55.2±0.8
KMedoid	79.3±11.8	58.3±10.1	93.3±13.7	100.0±0.0	80.0±9.2	83.7±7.0	87.0±14.9	OOM
KMeans	100.0±0.0	90.0±8.4	100.0±0.0	96.9±5.5	82.5±6.4	99.3±2.0	100.0±0.0	54.2±5.0
FFS	96.4±6.3	86.7±10.3	66.7±18.7	77.5±7.7	65.0±6.1	97.0±3.4	95.0±8.9	52.0±5.1
MC-RBF	71.4±0.0	81.7±5.1	66.7±0.0	75.0±0.0	60.0±0.0	52.0±4.6	85.0±8.9	28.1±4.0
MC-Cos	100.0±0.0	100.0±0.0	100.0±0.0	87.5±0.0	82.5±5.5	93.7±1.5	85.0±11.0	37.5±2.4
MinCUT	78.6±8.7	83.3±15.3	95.0±12.2	12.5±0.0	10.0±0.0	85.7±5.0	99.0±4.5	42.2±4.6
FeatProp	85.7±0.0	83.3±0.0	100.0±0.0	100.0±0.0	70.0±0.0	60.0±0.0	80.0±0.0	45.0±0.0
SDCN	85.7±9.5	86.7±10.5	93.3±14.0	82.5±8.7	67.5±10.4	72.7±9.1	87.5±10.4	56.7±2.9
EGAE	98.6±4.5	93.3±8.6	86.7±17.2	88.8±4.0	85.0±8.5	76.7±11.4	86.0±9.7	59.2±5.8
GCC	100.0±0.0	95.0±8.0	100.0±0.0	87.5±0.0	85.0±7.1	92.7±3.8	100.0±0.0	72.5±3.4
DMoN	90.0±9.4	90.0±10.0	91.7±14.8	87.5±5.7	76.0±9.9	90.3±4.6	97.0±7.3	60.2±6.3
RS-GNN	100.0±0.0	90.0±8.4	100.0±0.0	98.8±3.9	82.5±9.7	100.0±0.0	100.0±0.0	54.8±3.6



(a) (b) (c)

Figure 3: (a) A UMAP visualization of the node embeddings and the selected representatives for Cora (colors represent the class to which the nodes/representatives belong), (b) for CiteSeer, (c) RS-GNN results on Cora for different values of λ.

representative belongs. According to the visualization. the nodes from each class have formed one or more dense clusters and our model has selected a representative from (almost all of) these dense regions of points. More visualizations can be found in supplementary material.

A.5 Time Complexity and Memory Scalability

Let m represent the number of nodes, d represent the average degree of nodes, n represent the number of features (and, for simplicity, the embedding dimension), and k represent the number of representatives. The time complexity of each epoch in Algorithm 1 is goverened by $O(mnd)$ for computing graph convolusions, $O(mn^2)$ for computing node projections and bilinear functions, and $O(mnk)$ for assigning nodes to representatives. Overall, this gives a time complexity of $O(mn(d+n+k))$ for each epoch. For large datasets where a large number of representatives is needed (i.e. $k > n$ and $k > d$), the time complexity becomes $O(mnk)$. The memory complexity is $O(mk)$ for constructing and storing the matrix of distances from each node to its representatives.

When m and k are both large, Algorithm 1 may exhaust the accelerator memory due to its $O(mk)$ complexity. To reduce the memory usage and allow for applying RS-GNN to such settings, we modify Algorithm 1 to Algorithm 2 (in the Appendix). The main modifications include: 1- as is common in the GNN scalability literature (see, e.g., [37]), we replace the GCN modules with a

Table 4: Ablation study results for RS-GNN.

Ablation	Cora	CiteSeer	PubMed	Photos	PC	CS	Physics	Avg.
NoNorm	61.6±7.7	41.5±3.5	59.6±4.7	82.5±2.1	71.3±2.8	86.6±2.4	90.8±1.5	70.6
ConstNorm	59.3±6.9	44.3±4.2	61.7±3.9	84.2±1.4	75.0±3.1	86.7±1.3	90.5±2.1	71.7
Full Model	72.4±3.0	54.7±3.9	65.8±3.0	86.3±1.4	74.3±1.7	89.3±0.8	90.0±2.6	76.1

Table 5: Classification accuracies when a graph structure is not provided as input. Selecting $5c$ representatives. Bold numbers indicate statistically significant winner(s) following a t-test (p-value=0.05).

Selector	Model	Cora	Citeseer	Pubmed	Photos	PC	CS	Physics	Avg.
Random	MLP	41.9±2.9	37.4±4.0	51.9±5.0	57.5±4.0	55.2±4.4	76.1±2.7	76.7±5.2	56.7
Random	GCN	57.7±4.1	57.7±4.5	59.6±4.6	79.2±3.4	71.8±4.4	84.8±2.2	86.0±2.9	71.0
KMedoid	MLP	39.3±0.9	33.4±0.7	46.1±1.0	40.6±0.9	47.5±0.8	62.8±1.6	67.0±0.8	48.1
KMedoid	GCN	52.5±1.4	57.4±0.8	55.0±0.7	72.3±0.6	63.4±1.8	66.2±2.9	70.8±0.3	62.5
KMeans	MLP	42.4±0.9	40.1±1.1	58.5±1.3	70.0±1.0	63.7±0.8	68.5±0.7	77.4±0.3	60.1
KMeans	GCN	56.5±0.8	55.9±0.8	**72.7**±0.3	80.9±0.6	**75.8**±0.6	80.0±0.4	83.7±0.8	72.2
FFS	MLP	39.2±3.7	44.0±3.1	43.1±3.0	60.2±3.8	55.8±1.9	56.3±0.9	77.7±2.5	53.8
FFS	GCN	56.9±2.6	62.3±3.7	48.8±3.9	80.7±2.0	**74.8**±2.5	59.8±2.5	84.0±2.1	66.8
MC-Cos	MLP	46.5±2.2	47.5±2.3	52.1±0.7	54.0±2.4	58.2±1.2	83.0±0.5	86.8±0.7	61.2
MC-Cos	GCN	62.0±1.7	**63.0**±2.5	59.3±1.3	79.5±0.5	**75.4**±1.1	**87.6**±0.3	**92.8**±0.3	74.2
RS-GNN	GCN	**64.6**±2.4	**64.3**±2.1	65.1±3.0	**82.2**±2.0	**75.5**±2.1	**88.3**±1.6	89.9±1.9	**75.7**

SGC module [108] and pre-compute the graph convolutions $\boldsymbol{F}$ for the original graph, 2- we create corrupted graphs *nCorrupt* times and pre-compute the graph convolutions $\boldsymbol{F}'$ for the corrupted graphs, 3- at the beginning of each epoch, we randomly select a subset $\boldsymbol{F}''$ of $\boldsymbol{F}'$ to make the size match that of $\boldsymbol{F}$, 4- we batch the data and compute the loss and gradients for each batch separately and then aggregate the gradients and update the parameters. Assuming the batch size is b, the memory complexity for Algorithm 2 reduces to $O(bk)$ as each batch can be computed separately. Moreover, Algorithm 2 is also amenable to parallelization on multiple accelerators by distributing the batches across the accelerators.

Note that computing the loss separately for each batch corresponds to approximating $\boldsymbol{s}$ and $\boldsymbol{\mu}$ with the batch data as opposed to the entire data, but the approximation is expected to be close to the true value if the batch sizes are large enough. To this end, the batch size can be set to the largest number that does not exhaust the memory. We tested the Algorithm 2 version of RS-GNN on the Arxiv dataset when setting *nCorrupt* to 10 and $b = m/8$ (distributing over 8 accelerators). In terms of performance, we obtained an accuracy of 52.9 ± 1.6 (compared to 52.6 ± 1.2 for the original algorithm) showing that the approximations do not result in a performance degradation.

A.6 Ablation Study: CenterNorm and the Value of λ

We conduct an ablation study to verify the role of CenterNorm. To ablate CenterNorm, we run our model under two settings: (1) we do not normalize (we refer to this as "NoNorm"), and (2) we divide all the values in the embedding matrix by a constant number corresponding to the mean of the ℓ_2-norms of the embeddings (we refer to this as "ConstNorm"). According to the results in Table 4, our model benefits from CenterNorm and CenterNorm is more effective than ConstNorm because the per-node ℓ_2 normalization of CenterNorm helps bring the nodes within one cluster closer to each other which helps in identifying clusters and selecting representatives.

To verify the sensitivity of RS-GNN to the value of λ used in equation **??**, we ran RS-GNN on Cora with different values for λ. The results are presented in Figure 3(b). When $\lambda = 0$, the representatives will not receive gradients and hence the model ends up selecting random representatives. Therefore, the accuracy is quite low. For non-zero values of λ, we observe that the model is not highly sensitive to the value of λ and achieves good results for values in a large range. The model reaches its highest performance around $\lambda = 0.001$, and then the performance starts to slightly decrease for larger values of λ.

Table 6: Normalized mutual information (NMI) scores between the ground-truth labels of nodes and their cluster assignments. We take the results of the baselines from [100]. Bold numbers indicate the winners. We use — to indicate that the results were not reported (either because the model did not converge, or because the model did not scale to the dataset, or because the model was not tested on the dataset).

Selector	Context	Cora	CiteSeer	Pubmed	Photos	PC	CS	Physics	Avg.
KMeans	$\mathbf{X}$	18.5	24.5	19.4	28.8	21.1	35.7	30.6	25.5
SBM	$\mathbf{A}$	36.2	15.3	16.4	59.3	48.4	58.0	45.4	39.9
MinCut	$\mathbf{X}, \mathbf{A}$	35.8	25.9	25.4	—	—	64.6	48.3	—
AGC	$\mathbf{X}, \mathbf{A}$	34.1	25.5	18.2	59.0	**51.3**	43.3	—	—
DAEGC	$\mathbf{X}, \mathbf{A}$	8.3	4.3	4.4	47.6	42.5	36.3	—	—
SDCN	$\mathbf{X}, \mathbf{A}$	27.9	31.4	19.5	41.7	24.9	59.3	50.4	36.4
NOCD	$\mathbf{X}, \mathbf{A}$	46.3	20.0	25.5	62.3	44.8	70.5	51.9	45.9
DMoN	$\mathbf{X}, \mathbf{A}$	48.8	33.7	**29.8**	**63.3**	49.3	69.1	51.9	49.4
RS-GNN	$\mathbf{X}, \mathbf{A}$	**55.4**$\pm$0.8	41.3$\pm$1.0	26.1$\pm$3.6	58.3$\pm$1.6	50.1$\pm$0.9	**75.8**$\pm$1.2	**56.9**$\pm$3.3	**52.0**

Attributed Graph Clustering: RS-GNN can also be used for attributed graph clustering: we cluster the nodes in each dataset into c groups (recall that c is the number of classes) by assigning each node to its closest representative. While our main motivation is RSL, for completeness we show the performance of our model for attributed graph clustering as well. We note that different works on attributed graph clustering use different settings that are not directly comparable (e.g., some works use the same hyperparameters for all datasets, whereas some other works optimize the hyperparameters for each dataset and report the best test performance). Since our setting is similar to that of [100] and the results for many models have been provided in that work, we compare RS-GNN against the model proposed in [100] and their baselines. This includes KMeans, SBM (this works by estimating [90] a constrained Stochastic Block Model [99] with given number of k clusters), MinCut, AGC [119], DAEGC [104], SDCN, NOCD [96], and DMoN [100].

Table 6 shows a comparison of RS-GNN with the baselines in terms of the normalized mutual information (NMI) score (an established score for measuring and comparing clustering algorithms) between the cluster assignments and the node labels. From the results, we can observe that RS-GNN also shows a good performance for attributed graph clustering.

B More Related Work

Other works that are related to our work can be grouped as follows.

Feature selection: RS and feature selection are transposed views of a similar problem when it comes to compressing or summarizing datasets. Both have been studied extensively via *filter* and *wrapper* methods: an evaluation based on final task performance or on some proxy metric such as correlation, redundancy, coverage (or more general submodular functions), or the distance of selected entities [73, 86, 13, 9] as well as their mutual information or correlation with the prediction labels [29, 85]. While these methods tackle the diversity of the sample set [1, 57, 116, 10], there has also been extensive attention on taking fairness constraints into account as well [69, 92, 72, 98, 6].

Supervised data subset selection: Given a large dataset of labeled training examples, a class of RS models aim at selecting a small representative set from the dataset to reduce training time without substantially sacrificing model accuracy (see, e.g., [65, 64, 105, 63, 30, 88, 83]). For example, [65] aim at selecting a set of training data points such that a model trained on these examples generalizes well to the validation set and [83] aim at selecting a set of training data points whose gradients approximate the gradient of the full dataset. These models have been also applied to mini-batch active learning where a small set of labeled data points are assumed to be initially provided and then the next batches are selected based on pseudo-labels predicted by the model trained on the data available so far. While these models assume the data labels are available when selecting representatives, in this paper we assume no labels are provided as input.

Graph pooling: A technique commonly used in graph representation learning (especially for learning a representation for the entire graph) is graph pooling [75], where the nodes of the graph are iteratively

coarsened into "super-nodes". The parameters for the pooling operation are trained with the rest of the model parameters either to minimize a supervised loss (e.g., graph classification) or an unsupervised loss (e.g., graph reconstruction) [40, 43]. Graph pooling techniques can be classified into two categories: 1- clustering pooling (these are in the same vein as the graph clustering algorithms discussed earlier) and 2- node drop pooling. Clustering pooling approaches [113, 70, 115, 3, 11, 77] employ a differentiable graph clustering algorithm and consider each cluster to be a super-node; these approaches can be re-purposed for RS by selecting the node closest in the latent space to each super-node as a representative. Node drop clustering approaches [40, 74, 87, 117, 42] operate by dropping unimportant nodes and retaining the important nodes as the super-nodes. Graph pooling approaches have been mostly developed for smaller-sized graphs such as molecules that exhibit specific properties. For example, many of these approaches employ a GNN that assigns importance scores to each node, and then select the top-k most similar nodes. Such an approach assigns similar importance scores to highly similar nodes and results in sampling only from some parts of the graph. While this might be a reasonable approach for molecule classification tasks (as it makes the model focus on a few important sub-structures), it may not select a subset of the nodes that cover the entire graph (which is the desired property in our work). Nevertheless, in our experiments, we compare against several graph pooling approaches from both categories, both for RS and clustering tasks.

C CenterNorm Motivation

With the joint loss function used in the main text, the model can trivially reduce $\mathcal{L}_{\mathsf{SEL}}$ by making the values in the embedding matrix $\boldsymbol{H}$ arbitrarily small. That is because multiplying a small constant to $\boldsymbol{H}$ may not change $\mathcal{L}_{\mathsf{EMB}}$ substantially, but it can make the distances between the nodes arbitrarily small, resulting in a low value for $\mathcal{L}_{\mathsf{SEL}}$ even for a random representative embedding matrix $\boldsymbol{R}$.

One way to avoid the aforementioned problem is by normalizing the embedding matrix $\boldsymbol{H}$ before using it for selection. However, one should be careful about the choice of the normalization to avoid losing useful information. Before explaining how we normalize the embeddings, we describe a property of DGI embeddings that motivates our normalization.

Figure 4(b) shows a UMAP plot of the DGI embeddings $\boldsymbol{H}$ for the nodes in the original graph of Cora and the embeddings $\boldsymbol{H}'$ for the nodes in a corrupted Cora graph (red). The $\boldsymbol{H}'$ embeddings form a large cluster that is mostly in between the $\boldsymbol{H}$ embeddings and the $\boldsymbol{H}$ embeddings form small size clusters that are placed around the large cluster of $\boldsymbol{H}'$. To understand why this happens, notice that when we shuffle the node features for creating corrupted graphs for DGI training, the node features of the neighbors of each node are a random subsample of the node features in the graph. Therefore, the GNN aggregation function applied on the projected node embeddings makes the embeddings go toward the mean of projected embeddings. This makes the corrupted node embeddings $\boldsymbol{H}'$ form a large cluster in the middle and the embeddings $\boldsymbol{H}$ be placed outside and around this cluster.

Besides visual inspection, we also cluster the node embeddings $\boldsymbol{H}$ for Cora into 7 clusters using KMeans (7 is the number of classes in Cora; this provides good clusters with a normalized mutual information of 55.95 with the node labels). Then we compute the distance between each cluster center and the mean of these centers and obtain the following seven distances: 1.4, 1.2, 1.3, 1.2, 0.6, 1.4, 1.5. All cluster centers are at a good distance from the mean and, with the exception of one cluster, they are at a similar distance from the mean. We then subtract the mean and compute the angle between the cluster centers. We observe that the minimum angle between two cluster centers is 60.1 degrees and the average angle is 76.6 degrees.

The above analysis motivates the CenterNorm normalization outlined in the main text. Furthermore, the analysis shows that DGI is a good candidate for RSL as it groups data points into small-sized dense clusters in the latent space, thus an RSL algorithm can select representatives from each of the dense clusters.

D Proof of the Theorem

Theorem D.1. *There is no polynomial-time representative selection algorithm for FoN with an approximation factor better than $\omega(n^{-1/\mathrm{poly}\log\log n})$, unless the ETH fails.*

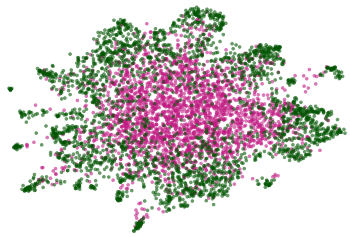


Figure 4: A UMAP plot of the DGI node embeddings for the nodes in the original graph of Cora (green) and the nodes in a corrupted Cora graph (red).

We start with two lemmas before proceeding to prove this result. We use (i, j) to represent a data point in the FoN problem with value 1 on features i and j.

Lemma D.2. *Let $\mathcal{S}$ be the set of data points selected by an RSL algorithm. Let (i, j) be a data point such that for all t we have $(i, t) \notin \mathcal{S}$, or for all t we have $(t, j) \notin \mathcal{S}$. Then the label of (i, j) is independent of the labels of $\mathcal{S}$.*

Proof. Let $\mathcal{S}$ be the set of data points selected by an RSL algorithm. Let (i, j) be a data point such that for all t we have $(i, t) \notin \mathcal{S}$. This means that the labels of data points in $\mathcal{S}$ are independent of the type of feature i. Recall that the type of feature i is chosen independently and uniformly at random. Hence, conditioned on the labels in $\mathcal{S}$, the label of (i, j) is 0 with probability $1/2$ and 1 with probability $1/2$. Similar argument holds when for all t we have $(t, j) \notin \mathcal{S}$. □

Lemma D.3. *Let $(i_0, i_1), (i_1, i_2), (i_2, i_3), \ldots, (i_{l-1}, i_l)$ be a sequence of data points such that for all $t \in \{1, \ldots, l\}$ we have $(i_{t-1}, i_t) \in \mathcal{S}$. Given the labels of the data points in $\mathcal{S}$ we can infer the label of (i_0, i_l).*

Proof. Consider two data points (i, j) and (j, t). If the labels of both data points are 1, then the features i, j and t have the same type. Hence, the label of (i, t) is 1 too. If the labels of both of them are 0, then the type of features i and j are different, and the type of features j and t are different. Hence, the type of features i and t are the same, which means the label of (i, t) is 1. A similar argument shows that if either (i, j) or (j, t) has label 1 and the other has label 0, then the label of (i, t) is 0. Thus, knowing the labels of (i, j) and (j, t) determines the label of (i, t). Applying this inductively proves the lemma. □

Proof of Theorem 4.2. The proof goes by reducing the densest k-subgraph problem to FoN. In the densest k-subgraph problem, we have an unweighted graph $\mathbb{G}$, and the goal is to find a subgraph of $\mathbb{G}$ with k vertices and the maximum number of edges. We say an algorithm is an α-approximation algorithm for the densest k-subgraph problem if it returns a subgraph with k vertices where the number of edges is at least α times that of the densest k-subgraph. It is known that there is no $\omega(n^{-1/\text{poly}\log\log n})$-approximation polynomial-time algorithm for the densest k-subgraph problem unless ETH fails [80].

Next, we show how to transform an input of the densest k-subgraph problem to an input of FoN, and then show how to transform an approximate solution for FoN to an approximate solution for the densest k-subgraph problem while only increasing the approximation factor by a constant. Therefore an $\omega(n^{-1/\text{poly}\log\log n})$-approximation polynomial-time algorithm for the FoN implies an $\omega(n^{-1/\text{poly}\log\log n})$-approximation polynomial-time algorithm for the densest k-subgraph problem, which does not exist unless ETH fails.

Let $\mathbb{G} = (V, E)$ be an input to the densest k-subgraph problem.[6] For each vertex in $\mathbb{G}$ we define a feature and for each edge in $\mathbb{G}$ we construct a data point. For each data point corresponding to an edge (u, v), the value of the features corresponding to vertices u and v are 1 and the value of all other features are 0. As defined in the FoN problem the type (red or blue) of each feature is chosen independently and uniformly at random.

[6]Note that graph $\mathbb{G}$ is not an attributed graph, rather a simple graph which is an input to the densest k-subgraph problem.

Let $\mathbb{H} = (V_{\mathbb{H}}, E_{\mathbb{H}})$ be a densest k-subgraph of $\mathbb{G}$ and let $\mathbb{F}$ be a maximal spanning forest of $\mathbb{H}$. Note that since there is no cycle in $\mathbb{F}$, the number of edges in $\mathbb{F}$ is at most $k-1$. Moreover, since $\mathbb{F}$ is a maximal forest of $\mathbb{H}$, for each edge e in $\mathbb{H}$, there is a path between the endpoints of e in $\mathbb{F}$ (otherwise we could add e to $\mathbb{F}$). Hence, if we query the data points corresponding to the edges of $\mathbb{F}$, by Lemma D.3, we can determine the label of all the edges in $\mathbb{H}$, which is an $\frac{|E_{\mathbb{H}}|}{|E|}$ fraction of all data points. This gives us a solution with $\overline{\text{Acc}} \geq \Omega\big(\frac{|E_{\mathbb{H}}|}{|E|}\big)$.

Let $\mathcal{S}$ be the set of data points selected by an α-approximation RSL algorithm, and $V_{\mathcal{S}}$ be the set of vertices adjacent to the edges corresponding to the data points in $\mathcal{S}$. By Lemma D.2, if the edge corresponding to a data point has one (or two) endpoints in $V \setminus V_{\mathcal{S}}$, then the label of that data point is independent of the labels of $\mathcal{S}$. Hence, the number of data points whose label is not independent of the labels in $\mathcal{S}$ is at most the number of edges induced by $V_{\mathcal{S}}$. We denote this edge set by $E_{\mathcal{S}}$. Recall that $\mathcal{S}$ is an α-approximate solution, i.e., $|E_{\mathcal{S}}| = \Omega(\alpha|E_{\mathbb{H}}|)$. On the other hand, $|\mathcal{S}| \leq k$ and hence $|V_{\mathcal{S}}| \leq 2k$. One can decompose the induced subgraph of $V_{\mathcal{S}}$ into $\binom{4}{2} = 6$ subgraphs each with k vertices, and pick the one with the maximum number of edges. This gives an $\Omega(\alpha)$-approximate solution to the densest k-subgraph problem. □

E Implementation Details

Baselines: For KMeans, we select the closest node to each cluster center as a representative. For FFS and MaxCover, we select representatives sequentially. In FFS , the next representative is the node farthest away (by Euclidean distance) from the closest representative in the current set. In MaxCover, the next representative is the node that increases the coverage of the non-selected nodes the most. Note that the sequential nature of FFS and MC makes them less amenable to parallelization. Also note that when we run surrogate function baselines using DGI embeddings as context, their selections are informed by both node features and the graph structure. For DMoN and MinCut models, we compute cluster centers by averaging the node embeddings with respect to the (hard) cluster assignments, and then select the closest point to each cluster center as a representative.

Algorithm 2 Memory-Efficient RS-GNN.

Input: $\mathcal{G} = (\mathcal{V}, \boldsymbol{A}, \boldsymbol{X})$, k

Initialize $\boldsymbol{R}$, $\boldsymbol{\Theta}$, and $\boldsymbol{U}$
$\boldsymbol{F} = \mathsf{GC}(G)$, $\boldsymbol{F}' = []$
for i=1 **to** nCorrupt **do**
  $\mathcal{G}' = (\mathcal{V}, \boldsymbol{A}, \mathsf{shuffle}(\boldsymbol{X}))$
  $\boldsymbol{F}' = \mathsf{concat}(\boldsymbol{F}', \mathsf{GC}(\mathcal{G}'))$
for epoch=1 **to** #epochs **do**
  $\boldsymbol{F}'' = \mathsf{subsample}(\boldsymbol{F}', \mathsf{len}(\boldsymbol{F}))$
  $\nabla = \boldsymbol{0}$
  for $\boldsymbol{F}^{(b)}, \boldsymbol{F}^{(b'')} \in \mathsf{batch}(\boldsymbol{F}, \boldsymbol{F}'')$ **do**
    $\boldsymbol{H} = \boldsymbol{W}\boldsymbol{F}^{(b)}$, $\boldsymbol{H}' = \boldsymbol{W}\boldsymbol{F}^{(b'')}$
    Compute $\mathcal{L}_{\mathsf{EMB}}$ based on $\boldsymbol{H}$ and $\boldsymbol{H}'$
    $\boldsymbol{\mu} = \frac{1}{n}\sum_i \boldsymbol{H}_i$, $\boldsymbol{\zeta} = \|\boldsymbol{H} - \boldsymbol{\mu}\|$
    $\tilde{\boldsymbol{H}} = \mathsf{CenterNorm}(\boldsymbol{H}) = (\boldsymbol{H} - \boldsymbol{\mu})/\boldsymbol{\zeta}$
    $\mathcal{L}_{\mathsf{SEL}} = \sum_i \mathsf{min}_j \mathsf{Dist}(\tilde{\boldsymbol{H}}_i, \boldsymbol{R}_j)$
    $\mathcal{L} = \mathcal{L}_{\mathsf{EMB}} + \lambda\mathcal{L}_{\mathsf{SEL}}$
    Compute gradients for $\mathcal{L}$ and add to ∇
  Update parameters based on ∇
Let $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{H}}$ be the representative and normalized node embeddings with minimum $\mathcal{L}$ during training.
for j=1 **to** k **do**
  The j^{th} representative = $\mathsf{argmin}_i \mathsf{Dist}(\hat{\boldsymbol{H}}_i, \hat{\boldsymbol{R}}_j)$

We implemented our model and the baselines in Jax/Flax [14, 52] and used the Jraph library [44] for our GNN operations. Our experiments were done on a JellyFish TPU for all datasets except for the Arxiv dataset where we used a DragonFish TPU as the experiments with the Arxiv dataset require more memory. For our DGI model, we used a single-layer GCN model with SeLU activations [68]. For the experiments that had access to the original graph structure, we set the DGI hidden dimension to 512 for all datasets except for the Arxiv dataset where we set it to 256 to reduce memory usage. For the experiments with no access to the original graph structure, we set the DGI hidden dimension to 128 as there exists less signal in this case. We trained the DGI models for 2000 epochs both for our model and the baselines. For KMeans and KMedoid, we used the implementation in scikit-learn [89] and scikit-learn-extra[7] respectively. To reduce the quadratic time complexity of MaxCover, we apply MaxCover on a k-nearest neighbors similarity graph in the input features/embeddings as opposed

[7] https://github.com/scikit-learn-contrib/scikit-learn-extra

Table 7: Dataset statistics.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,278	1433	7
Citeseer	3,312	4,536	3703	6
Pubmed	19,717	44,324	500	3
Amazon Photo	7,650	119,081	745	8
Amazon PC	13,752	245,861	767	10
Coauthor CS	18,333	81,894	6,805	15
Coauthor PHY	34,493	247,962	8,415	5
OGBN-Arxiv	169,343	1,157,799	128	40

to the full graph. We used different hyperparameters for the RBF kernel (in the case of MaxCover with RBF similarities) and kNN and reported the values that resulted in the best overall accuracy across models. For MinCUT and DMoN, we used the implementation from the DMoN paper. For our model, we set λ in the main loss function to 0.001 for all datasets. Also, for the experiments where a graph structure is not provided as input, to create a kNN graph we connect each node to its closest 15 nodes for all the datasets. For the one-shot graph active learning models, unfortunately we did not find the code to be able to test the models in our setting. Therefore, we For the graph active learning baseline, unfortunately we did not find source codes to be able to test them in our setting. Therefore, we re-implemented FeatProp [109] and included the results of our implementation in the experiments. For SDCN, EGAE, and GCC, we used the public codes released by the authors to select representatives.

For the classification GCN model, we used a two-layer GCN model with PReLU activations [51] and with a hidden dimension of 32. We added a dropout layer after the first layer with a drop rate of 0.5. The weight decay was set to $5e^{-4}$. The GCN is trained based on the nodes in the selected set $\mathcal{S}$ of representatives. We randomly split the remaining nodes in $(\mathcal{V} - \mathcal{S})$ into validation and test sets by selecting 500 nodes for validation and the rest for testing.

We ran all the experiments 20 times (except for Arxiv where we ran it 10 times) with different random seeds and reported the mean and standard deviation of the runs. Our code will be released upon the acceptance of the paper.

F Datasets

We used eight established benchmarks in the GNN literature. A summary of our dataset statistics are provided in Table 7. The first three datasets are Cora, Citeseer, and Pubmed [94, 55]. These datasets are citation networks in which nodes represent papers, edges represent citations, features are bag-of-word abstracts, and the labels represent paper topics. The next two datasets are Amazon Photo and Amazon PC [97]. These two datasets correspond to photo and computers subgraphs of the Amazon copurchase graph. In these graphs, the nodes represent goods with an edge between two nodes representing that they have been frequently purchased together. Node features are bag-of-word reviews and class labels are product categories. The next two datasets are Coauthor CS and Coauthor Physics [97]. These are co-authorship networks for the computer science and physics fields based on the Microsoft Academic Graph respectively. The nodes in these two datasets represent authors, edges represent co-authorship, node features are a collection of paper keywords from author's papers, and he class labels are the most common fields of study. Our last dataset is OGBN-Arxiv [55] which is also a citation dataset similar to Cora, Citeseer, and Pubmed, but orders of magnitude bigger than the three. The features in this dataset are average word embeddings of the paper abstracts.

G Limitations

We identify the following limitations with our current work:

- Both our model and baselines optimize for micro-average classification accuracy; optimizing for macro-average classification accuracy may require extra terms in the loss function or architectural modifications.

- While optimizing for micro-average accuracy is common in various domains, it raises the risk of being unfair to smaller sub-populations by not selecting any representatives from them. One must be cautious when using our model or any other model that optimizes for micro-average accuracy in applications when such a fairness is important.