CS 395 Deep Learning
Project Proposal Report Format

Samuel Rosenblatt; Alex Trehubenko; Mako Bates
Spring 2020

1. Introduction

We will train a graph-convolution network (GCN) to classify graphs, using as its input only a small sub-graph taken by a form of graph sampling known as Respondent-Driven Sampling (RDS).

Many different phenomena can be modeled as graphs. Graph classification (conceptually no different from any other classification task) is a problem applicable to any such domain. For example, one could represent patterns of encrypted communication as graphs, and ask questions about the communities of people involved.

Because collecting or storing a “complete” graph datum or dataset is sometimes expensive or impossible, it is often advantageous to work with a sub-graph sampled from the larger graph and hope it is representative of the whole. RDS is a mature technique for collecting such samples.

We (will) demonstrate that a GCN trained and tested on such sub-graphs can make good classification judgments about the larger parent graphs. While the graph classification techniques we use are not novel, their application to sub-graphs is novel, and in this project we will investigate the sampling methods and feature generation techniques for the data, and the hyperparameters of GCN architecture to maximize accuracy for this new task.

2. Problem Definition and Algorithm

2.1 Task Definition

Our group is developing a neural net architecture that is effective at classifying small sub-graphs of networks that are gathered using RDS sampling. This classifier could prove useful in situations where sampling the network is costly or impossible. An example of costly node sampling is collecting information about a social network. Asking each person who they are connected to in the network and then sampling that person is an expensive task. Additionally, it is sometimes impossible to gather information about the full network in the case of hard to reach populations, such as a network of people who inject drugs or sex workers (Verdery et al., 2015; Heckathorn, 1997; Ghani et al., 1998; Costenbader et al., 2006)

In the case where it is advantageous or necessary to classify larger graphs via samples, we have a choice of how to sample the graphs. A simple random sample of nodes within the graph and their edges to other sampled nodes would technically produce a graph, and the categorical node attributes would approximate the population-level node attributes (in this case the population is the parent graph). However, if the subgraph is substantially smaller than the original, it will be more-or-less completely disconnected; it will fail to capture connectivity information like assortativity or transitivity.

One alternative sampling methodology is known as Respondent-Driven Sampling (RDS). This method of sampling is similar to Breadth-First Search (BFS) sampling (Kurant et al., 2011) except for a few differences. First, we may start our sample with more than one node, known as a "seed", in our case we began with five seed nodes. Second, as each successive "layer" is added to the sample ("layer" used in the same sense as with BFS here, instead of adding all neighbors of all nodes in the previous layer, you add *n* randomly chosen neighbors of each node from the previous layer (or less if there are not *n* neighbors to add). And third, as each node is sampled we assume we may query the node for node attribute data and edge data of the edges attached to that node.

This method of sampling, as with other random-walk-based sampling methods such as BFS produces biases in the sample of the nodes included due to network effects such as "the friendship paradox" which creates a positive correlation between the node a wave was sampled in and its degree(Heckathorn 1997; Kurant et al. 2011). However, by understanding how this bias is developed, we can use analytical methods to correct for it in our population estimates for the parent graphs. These methods are known as RDS estimators and will amount to graph-level features in our architecture (Heckathorn, 1997; Zhu et al., 2020; Tomas & Gile 2011; Chen & Lu, 2017).

In the preliminary works, we train a GCN on sub-graphs of networks representing either connections between drug users in Colorado springs, or high school friendships. The GCN takes a (sub) graph as input. Each node in each sample network contains the true degree of the node, and an indicator of if the node was an RDS starting point (called "seed node"). Each subgraph inherits the ground truth label of its parent. The output of the model is a label for each network. (Graph convolution recapitulates the starting graph structure; our current model builds per-node estimations of the overall label, which are averaged to get the overall output.) These outputs can be compared to ground truth labels in order to calculate the accuracy of the model.

2.2 Algorithm Definition

The baseline system is composed of two distinct steps. The first is using RDS to sample the original, larger, networks. This is necessary to create the testing and training datasets used by the GCN model. Next the GCN is trained.

The GCN used for the preliminary results is a basic implementation as described in (Wang). The GCN architecture consists of 2 convolutional layers, with ReLU activation between them. The first layer creates an intermediate representation of the sub-graphs with K learned attributes. For the intermediate architecture, K was set to 4. The next convolutional layer creates an output with node attributes one-hot-encoding the target labels. Using mean pooling, the label for the sub-graph is computed from the average prediction of its nodes. Our model is based on the code for the GCN tutorial which also describes this architecture (Wang). The weights for the model are adjusted using backpropagation with an Adam optimizer, and the loss is calculated using cross entropy loss between the predicted labels for the subgraphs, and the ground truth label. In the preliminary model we run for 20 epochs. This length seemed suitable since the classifier converges quickly. Additionally, the model will only save new weights if its performance on the withheld sub-graphs exceeds its previous best performance. Below is a pseudocode representation of the model.

As the project moves forward, the group intends to develop a deeper architecture that produces superior results to the GCN baseline. Ideally this architecture would be easily deployable to real examples of networks sampled with RDS in which less training information is available.

```
1. Identify and isolate testing and training ground truth graphs.
2. For each ground truth graph:
    a. For K samples per graph:
        i. RDS sample the ground truth graph.
            1. Begin with a small starting sample of seed nodes. In our
               case we use a random sample of size 5. But the starting
               sample need not be random and can be of any size and the
               RDS estimators will still asymptotically approach
               population values.
            2. While there are less than target number of nodes in sample,
               N (which we range between 5 and 20 in these results),
               select one of the nodes currently in the sample, v, and add
               up to $n$ random neighbors of v to the sample, recording
               node and edge data of v as we do so. In our case, we used a
               value of 3 for $n$.
            3. After the target number of nodes, N, have been added to the
               sample, use the edge data collected as each was added add
               the edges between all the nodes in our sample which
               appeared in the larger graph (i.e. create the subgraph H
               which is induced on the parent graph G using only the
               nodes collected in the sampling process described in step 2)
    b. Add the sub-graph to the appropriate test or train dataset. Include
       node-level attributes:
        i. The original degree of the node.
        ii. Whether or not the node was a seed in the RDS process.
3. Initialize GCN with arbitrary weights.
4. Best accuracy = 0
5. For 20 epochs:
```

```
a. Batch data into groups of 5 graphs
b. For each mini-batch:
    i. Apply the model to the batch to get an output vector of the label
       guesses:
        1. Send data to the first convolutional layer.
        2. ReLU activation layer
        3. Send data to second convolutional layer
        4. Set network classifications to the most common node
           classifications.
    ii. Adjust weights using back-propagation with cross entropy loss
        and Adam gradient descent.
c. Calculate validation set accuracy.
d. If validation set accuracy > best accuracy
    i. Save weights
    ii. Best accuracy = new test accuracy
e. Else:
    i. Do not save weights.
```

3. Experimental Evaluation

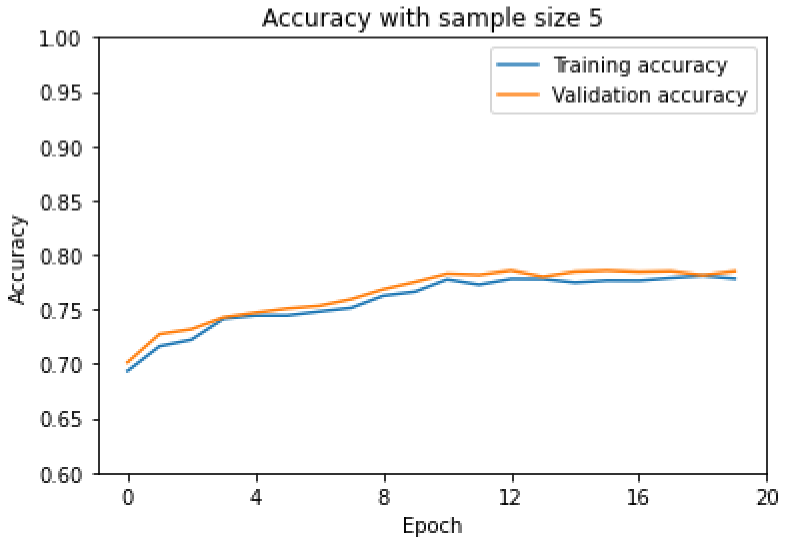
3.1 Methodology

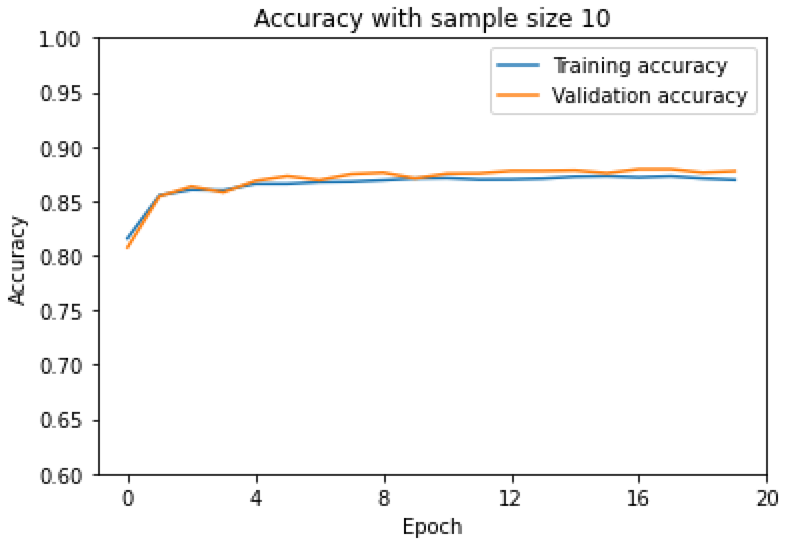
We believe (and, to a limited extent, have demonstrated) that a GCN can perform classification tasks on graph taking as input only small subgraphs sampled by RDS.

As in other classification tasks, the model's performance is its accuracy classifying sub-graphs. These validation samples are sub-graphs of parent graphs that are not used during training. We have already demonstrated that a simple classifier architecture can perform (much) better than random choice on an example dataset. In our next steps we will apply the same or larger models to more challenging classification tasks, and see if this kind of GCN is a generally applicable solution to the problem of sub-graph classification.

3.2 Results

Results indicate that the GCN can predict the class of sub-graphs reliably. After experimenting with the size of the RDS sub-graphs, it also becomes apparent that the classifier performs better with larger sub-graphs.





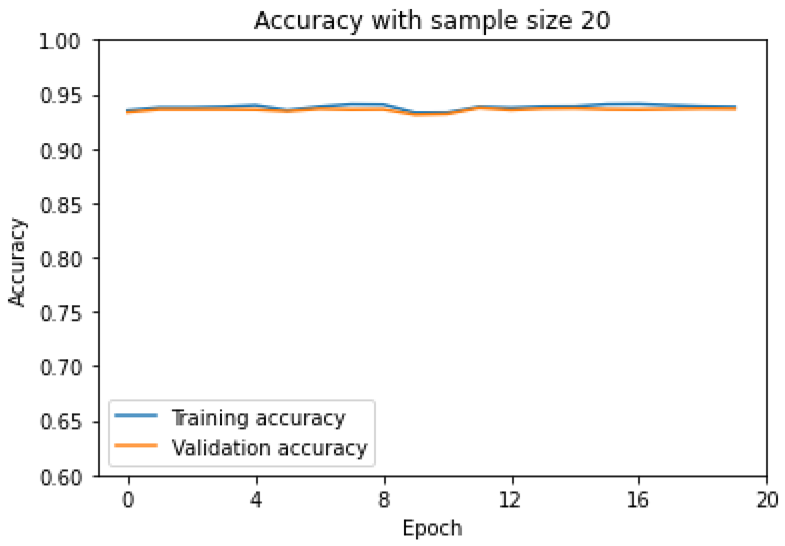


Figure 1: Accuracy of preliminary GCN with training and testing samples with varying size sub-graphs

3.3 Discussion

The accuracy of our baseline model, which was meant as a proof of concept, was surprising, and likely a result of the large intra-class distinctions, and small inter-class differences between the classes in our data due to each datum in each of the two classes being modeled after a single real-world network. Some of these differences, in reference to the node attribute we included in our algorithm, true degree, are shown in figures 2 and 3 below. However, our success does demonstrate the strengths of using RDS sampling as a part of the data pipeline for graph classification, as 77% accuracy using only 5 nodes in the sample would serve a purpose even for easy graph classification tasks, and it shows that depending on the data, this task can be completed with high accuracy.

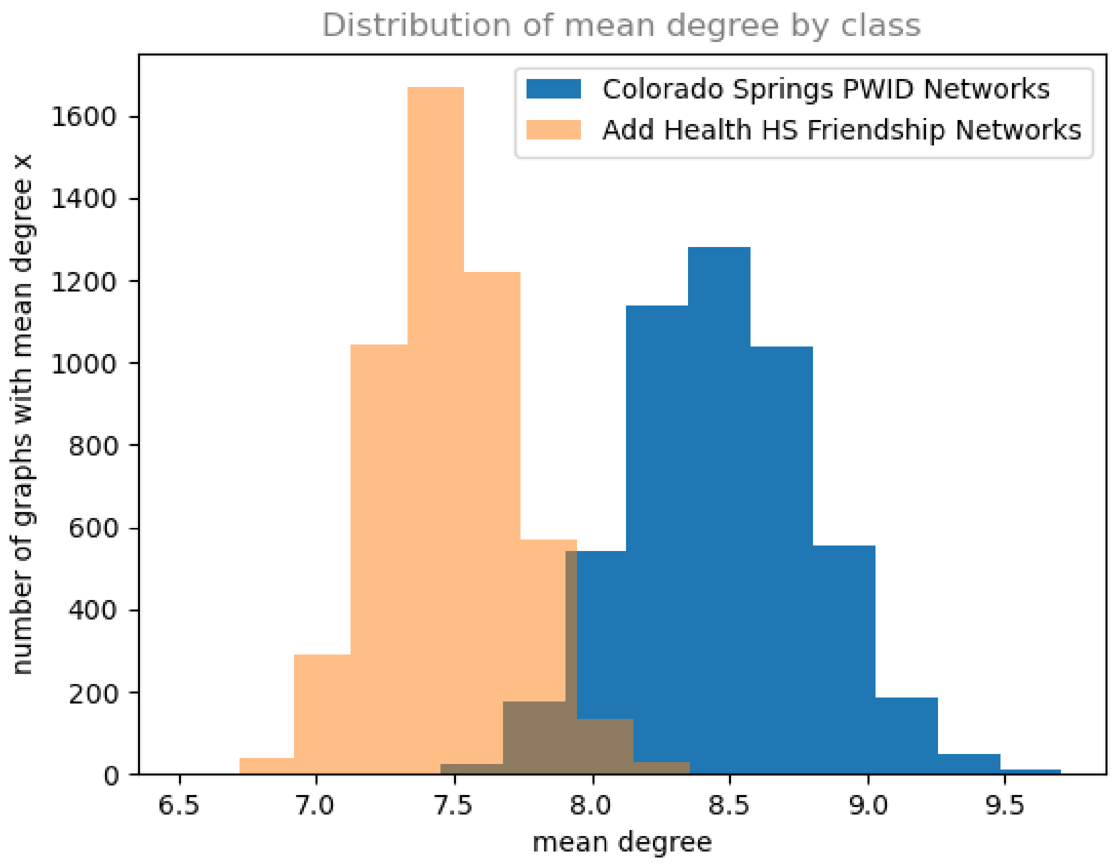


Figure 2: Distributions of mean degree for the two classes used in this dataset.

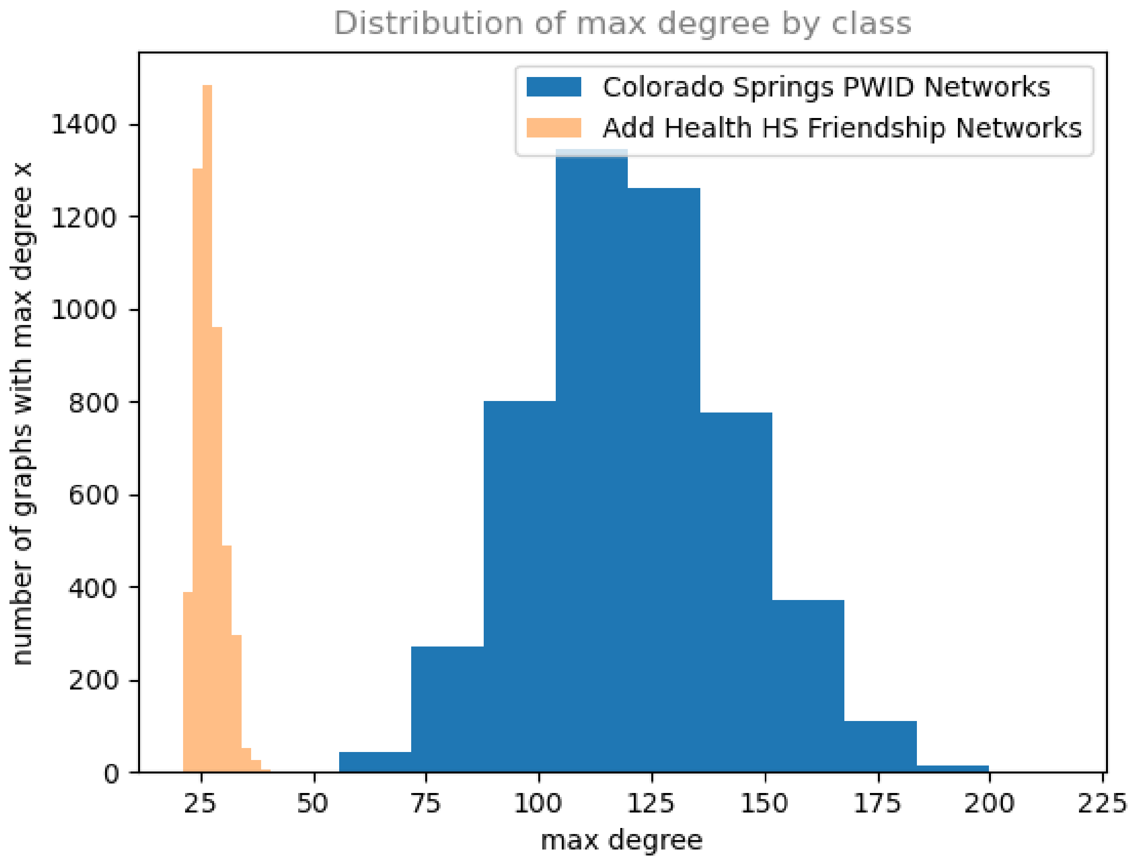


Figure 3: Distribution of maximum degree for each graph from the two classes used in this dataset

4. Related Work

RDS and its advantages and limitations are well studied. (Zhu et al. 2020) characterize the bias introduced in graph mean degree, homophily, and transitivity. (Wejnert 2009) also explores the induced bias of RDS, and shows that estimations of property variance are unreliable. Several other papers on this subject are listed in our bibliography. One of the main uses of RDS studies is to use RDS estimators to ascertain graph-level values about a population. Graph classification can be seen as a similar task to graph value estimation. Effectively, we would often like to know information about a graph, such as the value of its categorical variable, and only have access to a sample of said graph.

Graph convolution and GCNs have also been studied for several years. (Kipf & Welling) introduced the use of GCNs as graph classifiers. As an example application; (Zhou,Yao, Wu, *et al*) recently demonstrate the use of a crystal-chemistry-specific GCN system for identifying candidate battery materials.

The space of design decisions and architectural details that have been experimented with is quite large. DGL (https://docs.dgl.ai/en/0.4.x/index.html) is a mature PyTorch library for working with GCN, and it supports over a dozen layer types out of the box. Novel experimentation is ongoing; for example (Valsesia, Fracastoro, &Magli) demonstrate the viability of randomly-wired GCNs for enabling deeper architectures.

5. Next Steps

The preliminary works were simply intended to be a proof of concept, the accuracy of the classifier was a surprising result. Since the datasets are simulated, the true degree of the nodes was sufficient to classify the node well. Sub-graphs that are more similar to each other should increase the difficulty of the classification problem. Another way to increase the difficulty is introducing multiple classes. The three next steps are as follow:

1. Select new data. Several datasets are commonly used as benchmarks in graph classification literature. Papers with code provides a list of these datasets. We will select one or more of these datasets that have large graphs. This will serve the dual purpose of increasing the difficulty of the problem and allowing us to compare the accuracy of the sub-graph classifier to other graph classifiers.
2. We would like to incorporate three types of additional information to the model.
    a. Additional node information from the data can be added to the nodes tensor. For example the preliminary datasets include information on gender for each node which was not incorporated into the training in any way. The node level information that is available will depend on the new dataset that we choose.
    b. RDS tracks the distance of each node from the seed. This information can also be added to the nodes tensor. The distance from the seed is important since further away nodes are increasingly biased as each step of sampling increases exposure to high degree nodes.

    c. Graph level RDS estimators can be added. RDS-1 and RDS-2 are common RDS estimators for the proportion of nodes in the network that belong to a certain group (Wejnert). In the preliminary work this could be the proportion of each graph with each gender. In the next steps, this will again depend on which dataset we choose and its node level attributes.
3. Adjust the architecture of the classifier. The simple GCN architecture worked well in our preliminary results. However, as the difficulty of the problem increases the model might perform better with a higher dimension for the intermediate layer as controlled by the K hyperparameter. Another option to increase performance is increasing the depth of the model. We plan to heavily experiment with the architecture of the model to find a design that works particularly well for classifying sub-graphs.

6. Code and Dataset

The code for the model is available at
https://github.com/ShapeOfMatter/cs354_s21_project.
Data can be downloaded from
https://drive.google.com/drive/folders/16JSMGe493yxAs-LknEBv1sEEynKMhvHS?usp=sharing.

To replicate the paper begin by downloading the project code data from the links above. The data should be extracted into the project directory as follows.

- /med/original/[edgelists]
- /schoolnetJeffsNets/original[edgelists]

To run the project, install the necessary python version and packages either manually, or by running the install_dependencies.bash file in the repository.
The required packages are:

- Numpy
- Dgl
- Torch
- Networkx
- Matplotlib
- Glob
- Dataclasss-json
- Filelock

Most of the parameters in the model can be adjusting by changing values in the graph_training.settings file. The function of relevant settings are as follows:

- Max_batch_size: The maximum number of networks in a minibatch.
- New_data: Flag of whether or not to pull new graph samples. 0 = No new samples. 1 = New samples. Must create new samples the first time the code is run.
- Num_samples: The number of samples to be taken from each graph.
- Sample_size: The size of the subgraph.
- Epochs: The number of epochs.

To run the program call train_gcn.py with command line parameter 'graph_training.settings'. This parameter will point the program to the file which contains the settings.

7. Conclusion

In the preliminary results it is demonstrated that a GCN can accurately classify sub-graphs of our training data created with RDS. As expected, with larger sub-graphs, the GCN produces more accurate classifications. In the next steps of our project, a new dataset that increases the difficulty of classification will be introduced. This dataset will likely be one of the benchmark graph classification datasets mentioned in the previous section, which will allow for the results of this project to be compared to previous works. Two steps will be taken to address the increased difficulty. First, more node data and RDS estimators will be included in the networks that are being classified. Second, the architecture of the GCN will be modified and a hyper-parameter sweep will be performed. The authors intend to develop a final architecture that is particularly effective at classifying sub-graphs. Ideally, this architecture would be close to or above the benchmark accuracy of prior works that classify an entire network, despite using much less data. If this architecture is successful, it may prove useful in real world network classification problems where sampling is expensive.

Bibliography

Wang, Minjie, Quan Gan, Jake Zhao, and Zheng Zhang. “Graph Convolutional Network .” Graph Convolutional Network - DGL 0.4.3. Accessed March 31, 2021. https://docs.dgl.ai/en/0.4.x/tutorials/models/1_gnn/1_gcn.html.

Wejnert, Cyprian. “AN EMPIRICAL TEST OF RESPONDENT-DRIVEN SAMPLING: POINT ESTIMATES, VARIANCE, DEGREE MEASURES, AND OUT-OF-EQUILIBRIUM DATA.” *Sociological methodology* vol. 39,1 (2009): 73-116. doi:10.1111/j.1467-9531.2009.01216.x

Verdery, A. M., Merli, M. G., Moody, J., Smith, J., & Fisher, J. C. (2015). Respondent-driven sampling estimators under real and theoretical recruitment conditions of female sex workers in China. Epidemiology (Cambridge, Mass.), 26(5), 661.

Heckathorn, D. D. (1997). Respondent-driven sampling: a new approach to the study of hidden populations. Social problems, 44(2), 174-199.

Ghani, A. C., Donnelly, C. A., & Garnett, G. P. (1998). Sampling biases and missing data in explorations of sexual partner networks for the spread of sexually transmitted diseases. Statistics in medicine, 17(18), 2079-2097.

Costenbader, E. C., Astone, N. M., & Latkin, C. A. (2006). The dynamics of injection drug users’ personal networks and HIV risk behaviors. Addiction, 101(7), 1003-1013.

Kurant, M., Markopoulou, A., & Thiran, P. (2011). Towards unbiased BFS sampling. IEEE Journal on Selected Areas in Communications, 29(9), 1799-1809.

Zhu, L., Menzies, N. A., Wang, J., Linas, B. P., Goodreau, S. M., & Salomon, J. A. (2020). Estimation and correction of bias in network simulations based on respondent-driven sampling data. Scientific reports, 10(1), 1-11.

Tomas, A., & Gile, K. J. (2011). The effect of differential recruitment, non-response and non-recruitment on estimators for respondent-driven sampling. Electronic Journal of Statistics, 5, 899-934.

Chen, S., & Lu, X. (2017). An immunization strategy for hidden populations. Scientific reports, 7(1), 1-10.

Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. ICLR conference paper.

Linming Zhou, Archie Mingze Yao, Yongjun Wu, Ziyi Hu, Yuhui Huang, Zijian Hong, (2021). Machine Learning Enabled Prediction of Cathode Materials for Zn ion Batteries. https://arxiv.org/abs/2104.00586

Diego Valsesia, Giulia Fracastoro, Enrico Magli, (2021). RAN-GNNs: breaking the capacity limits of graph neural networks. https://arxiv.org/abs/2103.15565