**Introducing Diversity to gPool**

**CS 292F Project Report**

Marianne Arriola Ashley Bruce

[mariannearriola@ucsb.edu](mailto:mariannearriola@ucsb.edu) [ashleybruce@ucsb.edu](mailto:ashleybruce@ucsb.edu)

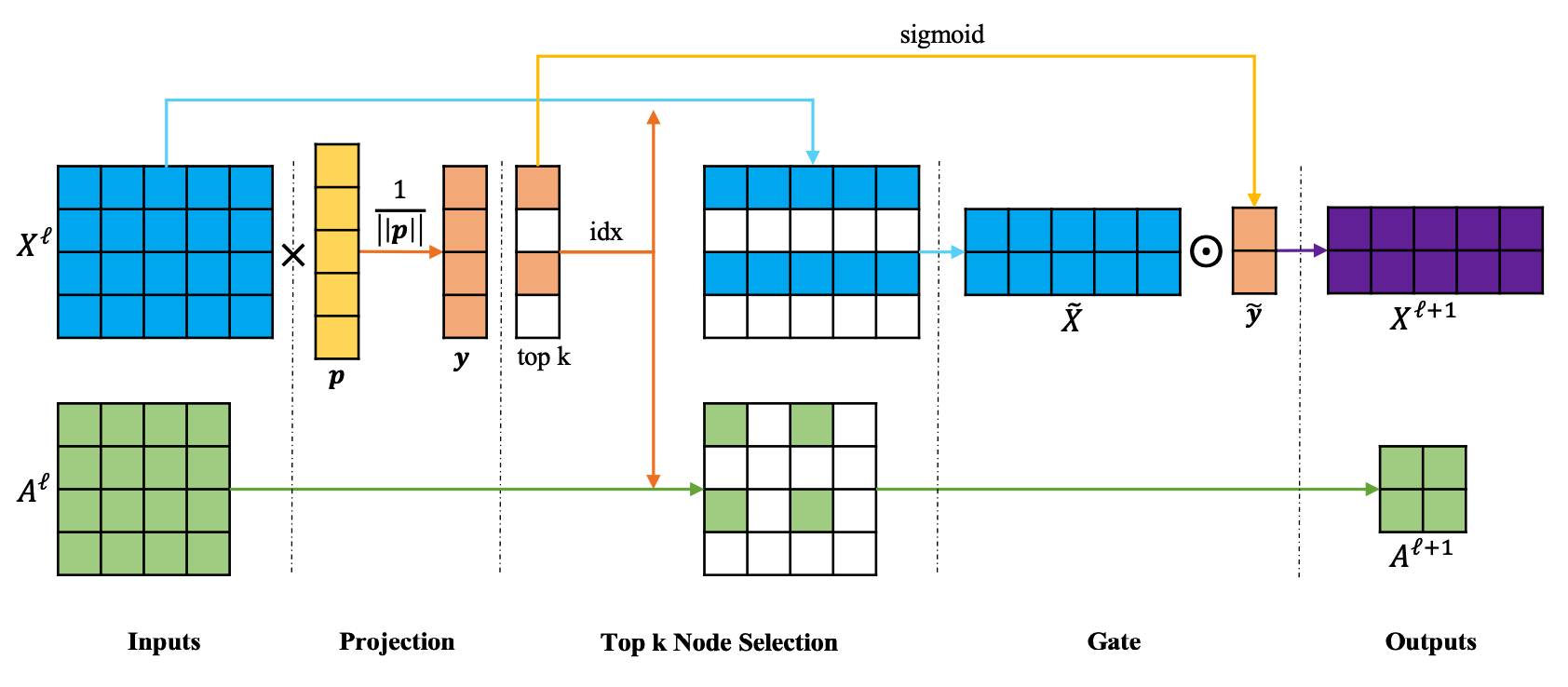
**1. ABSTRACT**

Graph U-Nets considers the problem of representation learning for graph data [1]. Its methods propose pooling and unpooling operations, where the pooling layers (gPool) adaptively select nodes to form a smaller graph based on their scalar projection that best represents the original graph. In only examining the projection values, gPool neglects to maintain diversity in the sparsified graph: in particular, ensuring that nodes selected originate from diverse neighborhoods in the original graph. Our work addresses this shortcoming and we improve gPool in Graph U-Nets by using the Jaccard Coefficient to rank nodes based on their similarity in neighborhoods with other nodes which is then used to weigh the scalar projection values. When choosing nodes from diverse neighborhoods for pooling, we found that graph classification performance can improve or worsen depending on the network structure.

**2. INTRODUCTION**

Convolutional Neural Networks are a class of artificial neural networks that is designed to learn spatial hierarchies of features through backpropagation. CNNs are typically composed of three types of layers: convolution, pooling, and fully connected layers [2]. The convolution and pooling layers perform feature extraction, while the final layer maps the extracted features into a final output, such as classification [2]. These neural networks are common among imaging data, but rarely seen applied to graph applications. This is due to the fact that the pooling and un-sampling operations for CNNs are not natural on graph data.

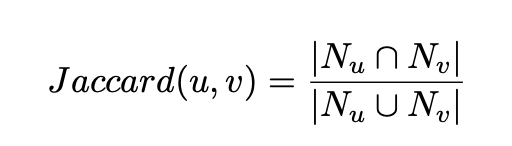
Graph U-Nets proposes novel graph pooling (gPool) and unpooling (gUnpool) operations to allow for CNNs to be applied to graphs, in particular, an encoder-decoder architecture for network embedding [1]. The graph pooling layer uses a trainable vector to map nodes from the original graph onto 1D space. gPool ranks nodes based on their scalar projection values to form a sparser graph that best represents the original graph for the following layer. gPool encodes high-level features for network embedding by reducing the size of the feature vectors. Once the features have been encoded, gUnpool restores the graph back to its original structure.



**Figure 1.** Graph pooling layer as outlined in original Graph U-Nets paper [1]. In this representation, k = 2, the original graph has 4 nodes and 5 features.

While gPool produces sparse graphs that preserve the most information from the original graph, it does not ensure that the sparse graph preserves its diversity.

In order to incorporate diversity into node sampling, we utilize the Jaccard Similarity Coefficient as a measure for node similarity in order to rank nodes based on their neighborhood similarity. This is a statistic to measure the similarity of sample sets and can be used to measure the similarity between two nodes based on their neighborhoods; it can be calculated utilizing the following formula, where Nu and Nv are the neighborhoods of nodes u and v, respectively.



**Figure 2**. Jaccard Similarity Coefficient Formula

The idea behind this measure is that the more similar the nodes are, the more likely they are to be within the same community. Therefore, by sampling nodes with low similarity scores, we can ensure that we are choosing nodes from different communities, and account for node diversity.

Since we are looking to select diverse nodes from the graph as a whole, we need to obtain similarity scores for all the nodes in the graph. Doing this requires us to compare each node to every other node, giving us a quadratic time complexity of O(n2 ).

**3. RELATED WORK**

When looking to increase node diversity, it is important to understand how this should be quantified. Applications of edge-betweenness centrality, such as the Girvan-Newman algorithm, divide graphs into multiple communities by removing edges by eliminating edges with high path lengths [3]. PageRank algorithms, which quantify the importance of “central nodes”, can be expanded to find local communities through random walks [4]. The Jaccard coefficient uses the neighborhoods of two nodes to calculate similarity scores, which can be used to determine if two nodes are a part of the same community [5].

There are many works that use node diversity in graphs for community detection specifically. In topics such as ecology or biology, being able to quantify and preserve diverse elements in a graph for preservation is important. Studies have been done to quantify this diversity and identify the components of the network that will maximize the number of diverse elements [6]. Novel families of graph neural networks have been proposed for solving community detection problems [7].

There have been a few papers that consider node diversity into their graph neural networks. Heterogeneous Graph Attention Networks considers the diversity of nodes when projecting a node’s features onto a feature space [8]. One graph representation learning model varies the degree of nodes sampled on the random walks to consider more diverse nodes in node embedding [9]. One graph neural network proposes a model of diverse sampling of different neighborhoods to improve the expressive power of GNNs [10].

To the best of our knowledge, there are no studies that examine the impact of enforcing node diversity in graph pooling methods.

**4. METHODS**

In the original architecture, nodes are assigned scores based on their projection onto a trainable vector and the “top-k” nodes are used for the graph in the next layer. We modify the node-ranking in the pooling operation in Graph U-Nets by using the Jaccard coefficient to rank nodes based on their neighborhood similarities.  
  
We assess the effectiveness of considering diversity in graph pooling in two ways: we rank nodes solely based on their Jaccard similarity and we rank nodes based on the scalar projection values that are weighted by values produced from ranking based on the Jaccard similarity.

For selection of nodes only based on Jaccard similarity scores, nodes were ranked according to their Jaccard similarity scores and the nodes with the lowest scores (i.e., nodes with the most dissimilar neighborhoods) were used for sparsification.

For selection of nodes based on a combination of both their neighborhood similarity and their projection score, nodes were first ranked based on their diversity: every node pair is given a score based on their Jaccard coefficient. As a higher Jaccard coefficient is associated with similarity, we aim to select nodes that have the lowest Jaccard coefficients such that selected nodes have dissimilar neighborhoods and therefore are diverse.  
  
We first score nodes based on their projection values then weigh the scores based on the diversity ranking such that scores of nodes that are selected for are given higher weight.  
  
Assigning static weights is not ideal, especially as diversity may be less or more important depending on the type of graph: for example, heterogeneity characterizes social networks, whereas homogeneity characterizes telecommunication networks. Instead of using static weights, we train them using gradient descent. In doing so, we also test whether selecting nodes by diversity improves prediction accuracy: if gradient descent assigns higher weights to nodes ranked by diversity, then this indicates that the diversity ranking lowers classification loss.   
  
To find the best weights, we use the graph classification loss for applying gradient descent. In particular, we implement noisy stochastic gradient descent: we select a random sample for loss calculation and add a noise value with mean 0 to the weight update. By adding noise, we prevent gradient descent from remaining in a saddle point.  
  
We test our methods on PROTEINS, a protein network dataset, and IMDBMULTI, an actor network. We select these datasets in particular because they are smaller in size in comparison to datasets like COLLAB and D&D, which is necessary as our approach is computationally expensive, and also to determine the robustness of our approach on networks from different contexts.

**5. EXPERIMENTAL RESULTS**

The graph classification accuracy from our experiments can be seen in the following tables. For our first experiment, we compared the original gPool’s accuracy to that calculated using just the Jaccard and a combination of Jaccard with gPool’s projection. The “k” nodes with lowest Jaccard scores had their projection scores multiplied by 0.9, while the scores of the rest of the nodes had their weights multiplied by 0.1. The results from this can be seen in the table below.

| **PROTEINS** | | | |
| --- | --- | --- | --- |
| **# Epochs** | **gPool with Projection** | **\*gPool with Jaccard** | **\*gPool with Projection + Jaccard** |
| 1 | 73.214% | 60.714% | **74.107%** |
| 10 | 74.107% | 75% | **75.893%** |
| 50 | 75.893% | **76.786%** | **76.786%** |

**Table 1.** Initial results comparing the original gPool to the use of the Jaccard coefficient. For gPool with both the projection and the Jaccard coefficient, nodes chosen by JC were multiplied by 0.9, while the nodes not chosen were multiplied by 0.1. \* Denotes our models.

As can be seen from the table, gPool that was performed using a combination of the projection scores and the Jaccard coefficient performed better across all epochs being tested for. As the weights being used (0.9 and 0.1) were chosen somewhat via “trial and error” we were curious if there was a way to train the weights being used to result in better accuracy. The following table depicts the results after implementation of a weight training algorithm.

| **PROTEINS** | | | | |
| --- | --- | --- | --- | --- |
| **# Epochs** | **gPool with Projection** | **\*gPool with Jaccard** | **\*gPool with Trained Weights** | **Weights** |
| 1 | 73.214% | 60.714% | **0.74107** | 0.5 and 0.5 |
| 10 | 74.107% | 75% | **76.786%** | 0.6301 and 0.3699 |
| 50 | 75.893% | **76.786%** | **76.786%** | 0.6301 and 0.3699 |
| **IMDBMULTI** | | | | |
| **# Epochs** | **gPool with Projection** | **\*gPool with Jaccard** | **\*gPool with Trained Weights** | **Weights** |
| 1 | **50.667%** | 48% | 50% | 0.5 and 0.5 |
| 10 | **55.333%** | 54% | 54% | 0.6744 and 0.3255 |
| 50 | **56.667%** | **56.667%** | 56% | 0.6892 and 0.3107 |

**Table 2.** Results comparing the use of a trainable weight vector to that of the original gPool and the use of only the Jaccard. Weights that resulted in the highest accuracy are recorded on the right. \* Denotes our models.

**6. DISCUSSION**

Results achieved by adding diversity to gPool methods varied by dataset used. For the PROTEINS dataset, we found that using a combination of the diversity ranking and the projection scores ranking results in network embedding that increases graph classification accuracy. As the classification accuracy increases and gradient descent trains the weights for nodes selected for diversity to be higher, we conclude that preserving diversity in the pooling operation leads to improved network embeddings in the PROTEINS dataset.

For the IMDBMULTI dataset, the original gPool with a trainable projection vector resulted in the best results. While the accuracy for gPool with the trained weights was not far behind that of the original, we believe that this could be because of the graph structure.

First, our method does not account for ‘noisy’ graphs: we select nodes from the most dissimilar neighborhoods in order to capture the graph diversity. However, it may be the case that an outlier node is selected that does not truly represent the community, and instead may be dissimilar from the others because it results from noise.  
  
Accounting for diversity may perform better on graphs with defined communities. Because our method aims to select nodes that represent their respective communities to promote diversity, if these communities are not distinct, then the node selection based on neighborhood similarity may be arbitrary (ex., a ring network, where there are no defined sub-communities). If the communities are well defined, then nodes that are more similar will be more likely to be correctly classified.

It is worth noting that the original Graph U-Nets paper ran 200 epochs on all four datasets. Our experiment only runs up to 50 epochs on two of the datasets. Calculation of the Jaccard coefficient happens in quadratic (O(n2)) time, and thus is very computationally expensive. Trying to compute the graph classification accuracy using the Jaccard for the two larger datasets from the original study, COLLAB and DD, resulted in 1 epoch taking upwards of an hour. We decided that the possible increase in graph classification accuracy was not worth the time cost, and excluded them from our study.

**7. FUTURE WORK**We hope to explore other diversity metrics for sparsification in gPool: for instance, instead of selecting nodes based on neighborhood diversity, we could also select nodes based on node diversity using the node features.

As it is computationally expensive to calculate each Jaccard coefficient for every node pair in the graph, we could explore different ways for ranking nodes based on neighborhood similarity. Instead of calculating the neighborhood similarity for every node, we can find a way to approximate it. While we did not officially measure how long it took for each model to finish running, we would include this if we explored similarity approximations to better determine their effectiveness.

For example, we could use random walks to determine which nodes are the most dissimilar in node neighborhoods: nodes that are farther down the random walk from an initial node are likely to come from dissimilar neighborhoods.   
  
We can further investigate the differences in classification accuracy between using PROTEINS and IMDBMULTI by running the datasets through community detection algorithms. This will allow us to determine whether the decrease in accuracy from applying our method to IMDBMULTI is due to a lack of clearly defined communities.  
  
We can also tune the ‘k’ value at each gPool layer, which determines how many nodes are included in the sparsified graph. Ideally, in the graph pooling, a node from each distinct community should be selected. We could use community detection to determine ‘k’ so that the sparsified graph contains a node from the ‘k’ communities in the graph.

We can account for noise by ensuring that selected nodes do not have neighborhood similarities that are drastically different from the majority. This will prevent ‘outlier’ nodes from being selected in the neighborhood ranking.

**8. REFERENCES**

[1]. Gao, H., Ji., S. (2019). Graph U-Nets. *International Conference on Machine Learning.*

[2]. Yamashitea, R., Nishio, M., Do, R., Togashi, K. (2018). Convolutional neural networks: an overview an application in radiology. *Insights into Imaging*. P.611-629.

[3]. Despalatovic, L., Vojkovic, T., Vukicevic, D., et al. (2014). Community structure in networks: Girvan-Newman algorithm improvement. *IEEE*.

[4]. Hollocou, A., Bonald, T., Lelarge, M. (2016). Improving PageRank for Local Community Detection. *arXiv*.

[5]. Wu, H., Chen, Y. (2020). Graph Sparsification with Generative Adversarial Network. *IEEE*.

[6]. Carpi, L., Scheber, A., Pardalos, P., et al. (2019). Assessing diversity in multiplex networks. *Scientific Reports*.

[7]. Chen, Z., Li, X., Bruna, J. (2017). Supervised Community Detection with Line Graph Neural Networks. *arXiv*.

[8]. Wang, X., Ji, H., Shi, C., et al. (2021). Heterogeneous Graph Attention Networks. *World Wide Web Conference*. P. 2022-2032.

[9]. Jeong, J., Yun, J., Keam, H., et al. (2020). div2vec: Diversity-Emphasized Node Embedding. *Proceedings of the ImpactRS*.

[10]. Anonymous authors. (2021). Towards Powerful Graph Neural Networks: Diversity Matters. *ICLR* (in review).