aMazeGNN: A Maze clustering GNN

Geena Kim*
geena.kim@colorado.edu

Paul William Falcone
pwfalcon@stanford.edu

Shan Lu
shanlu33@stanford.edu

1 Introduction

Visual navigation in robotics is an active research area where the goal is to use visual information to locate and navigate a robotic/autonomous agent. There have been an increasing number of approaches using deep learning for both visual information processing (localization) and navigation tasks, most of which use combinations of CNN, MLP, and RNN-family neural network architectures. Since the search space on state and action can be very large, a naive reinforcement learning approach does not always work well when navigating in a complex environment. By an abstraction of the environment to an abstraction graph, and then searching for a path in the abstraction graph can help navigating in a complex environment. Our project is about clustering on maze like grids to coarsen the graph and see if our algorithm can group similar room-like structures together This can be useful for robotic navigation in a new environment, by analyzing an unlabeled floor plan, the algorithm would be able to divide the layout into rooms, hallways, corridors, etc, which will then make it easier for it easier to perform further downstream tasks such as navigation. As the first step, we use a Graph Neural Network (GNN) to perform a k-means clustering on a maze graph. We show that a GNN can perform a k-means clustering on a 2D grid and suggest ideas to improve clustering on grids.

2 Data

We created a maze image and then applied pooling by 10 pixels to created a grid. Points in the grid are the nodes and edges are formed between the four (maximum) nodes up, down, left, and right to a given node. A pytorch dataset saves the raw grid (with the maximum node degree = 4) with coordinate as node feature and action tuples as edge features. The resulting graph has about 61 k nodes and 120 k edges.

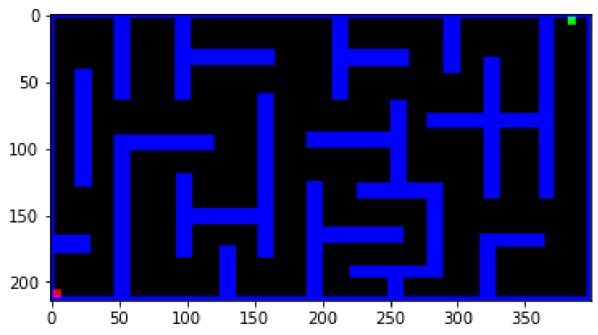

Figure 1: An example maze

3 Methods

For the clustering task, we use a 2-layer GraphSAGE architecture [1] with Euclidean loss and regularization terms. The final k-means loss is as follows:

$$\mathcal{L} = \overline{d_k^2} - \overline{d_{k'}^2} + \ell_{\text{sep}} + \ell_{\text{assign}} + \ell_{\text{num}} + \ell_{\text{inter}}$$

The first two terms are the mean squared distance from the nodes within the cluster k to the cluster mean $X_m^{(k)}$ and the mean squared distance from the nodes within other clusters $C_{k'}$ ($k \neq k'$). The first term incentivizes the nodes within the cluster close together and the second term incentivizes nodes from other clusters $C_{k'}$ far away from the cluster C_k. The cluster centroid is calculated as below, it is averaged by the cluster assignment probability, thus initially not an accurate centroid, but it converges to the real centroid overtime as assignment probabilities become more confident.
Soft centroid of the k^{th} cluster C_k is calculated:

$$X_m^{(k)} = \frac{\sum_i^N P_i^{(k)} X_i}{\sum_i^N P_i^{(k)}}$$

To get the cluster assignment probabilities, $P^{(k)}$, we apply the GNN to the graph and its features:

$$\text{GNN}(A, X) \rightarrow P^{(k)}$$

The ℓ_{sep} is a separation loss, which penalizes nodes from other clusters $C_{k'}(k' \neq k)$ being too close to the centroid of k-th cluster. The ϵ is a small constant to prevent divergence.

$$\ell_{\text{sep}} = \sum_k^K \sum_i^{N_{k'}} \frac{1}{((X_i^{k'} - X_m^k)^2 + \epsilon)}$$

ℓ_{assign} is a probability assignment loss which penalizes assignment probability of a node being equally distributed among the clusters.

$$\ell_{\text{assign}} \propto \sum_k^K \sum_i^{N_k} \frac{|P_i^{(k)} - 1/K|}{1/K}$$

ℓ_{num} is a loss that prevents all nodes being in single cluster.

$$\ell_{\text{num}} \propto \sum_k^K \frac{1}{\sum_i^N P_i^{(k)}} \approx \sum_k^K 1/N_k$$

The last loss term, ℓ_{inter}, is the sum of the mean squared distances between nodes within each cluster. It is a naive measure of the size of a cluster, and it further forces nodes in a cluster to be located close together.

$$\ell_{\text{inter}} = \sum_{k}^{K} \frac{1}{N_s^2} \sum_{i,j}^{s,s} (X_i^k - X_j^k)^2$$

, where s is a number of samples in the cluster k. For calculation efficiency we random sample 1000 nodes in each cluster.

4 Results

Using the k-means loss mentioned above and 2-layer GraphSAGE with Mean aggregator and four cluster assignments, we demonstrate the clustering performance of the model on the maze data. Fig. 2 shows the initial clustering assignment probability on train and test dataset, whereas Fig. 3 and 4 show the final assignment after 490 epochs of training. Comparing the figures 5-7 to the figures 2-4, one can see the different clustering behavior when with and without the ℓ_{inter} loss. When there is no ℓ_{inter} loss, some clusters get pushed away on the test data, and the training assignment patterns overlap, which means the model does not generalize well. With the ℓ_{inter} regularization term, the nodes in the cluster tend to group together and the prediction patterns on train and test data match.

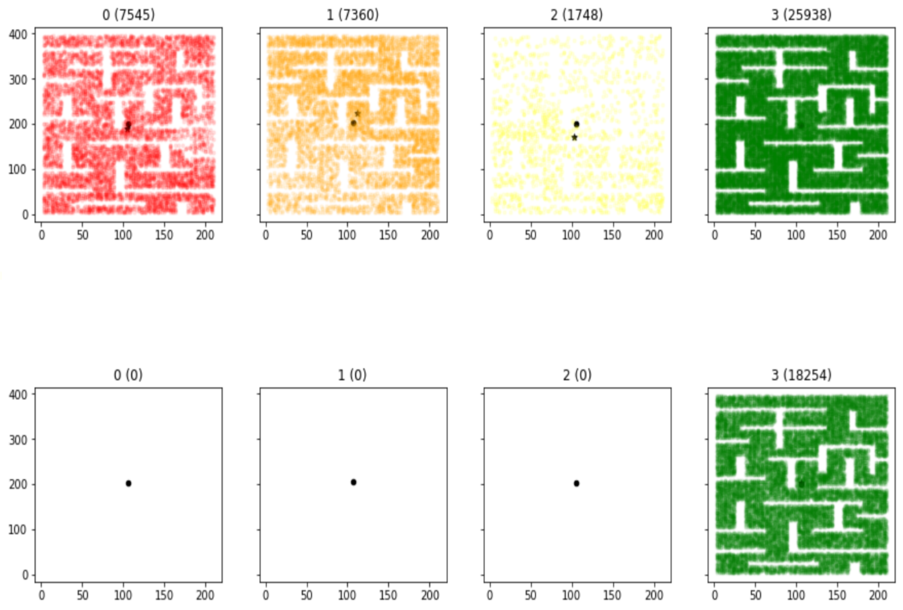

Figure 2: 4-cluster clustering at epoch 0. Upper four: on training dataset, Lower four: on test dataset. Titles indicate cluster index (Number of nodes in the cluster). Black circle marker is the soft centroid weighted by the probability assignment and black start marker is the hard centroid from $\max(P^k)$. The colored dots are the nodes assigned to each cluster.

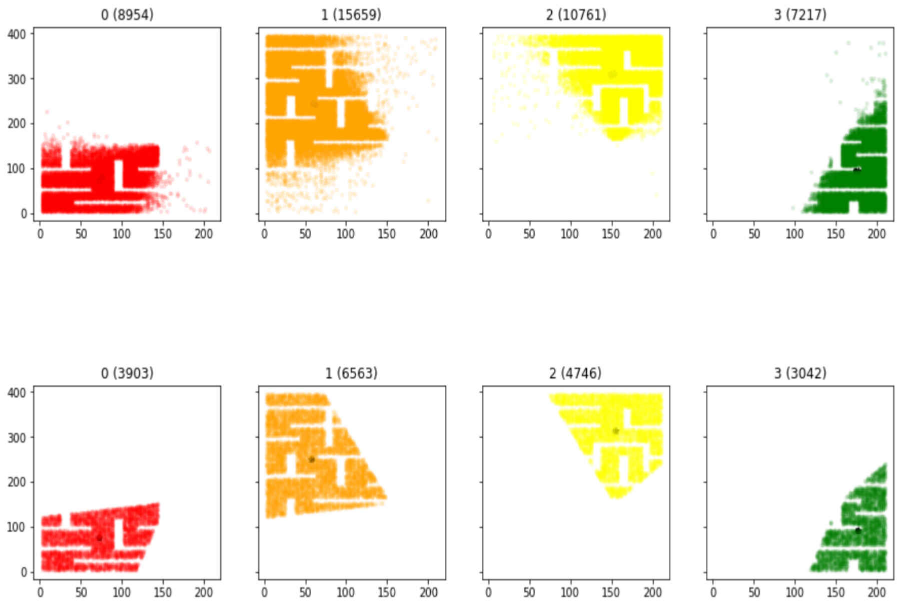

Figure 3: 4-cluster clustering at epoch 490.

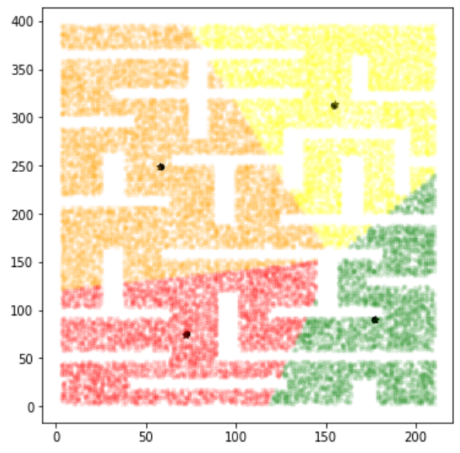
Figure 4: 4-cluster clustering at epoch 490, combined.

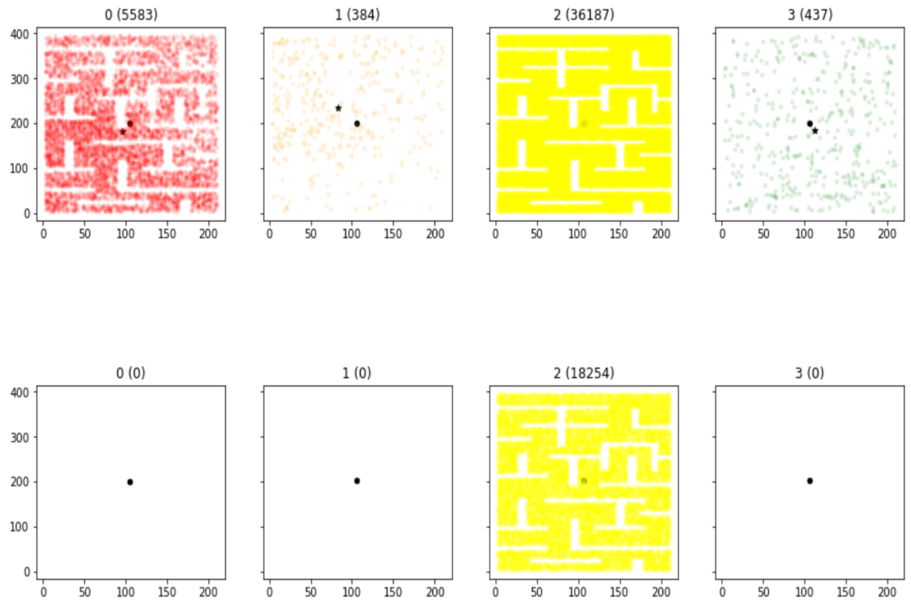

Figure 5: 4-cluster clustering at epoch 0 without ℓ_{inter}.

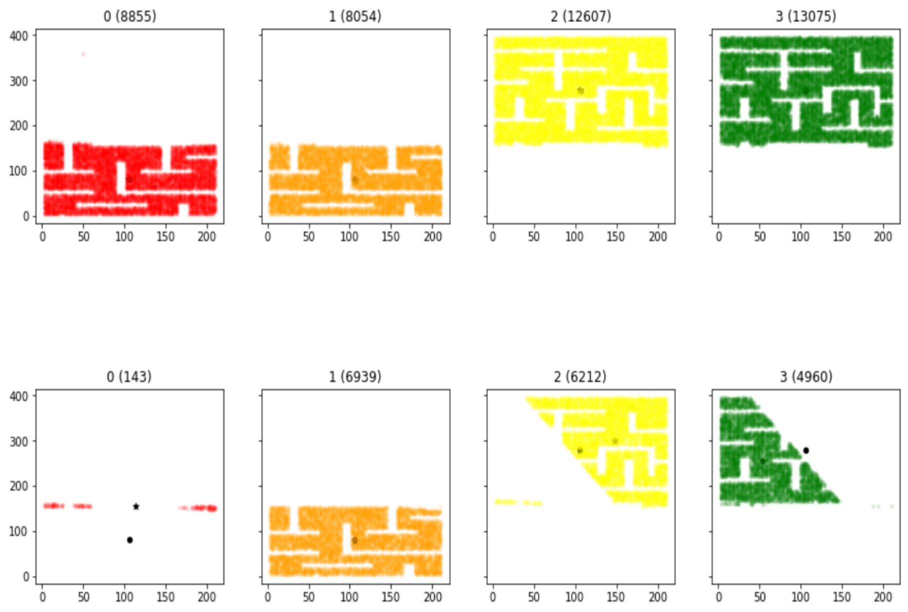

Figure 6: 4-cluster clustering at epoch 490 without ℓ_{inter}.

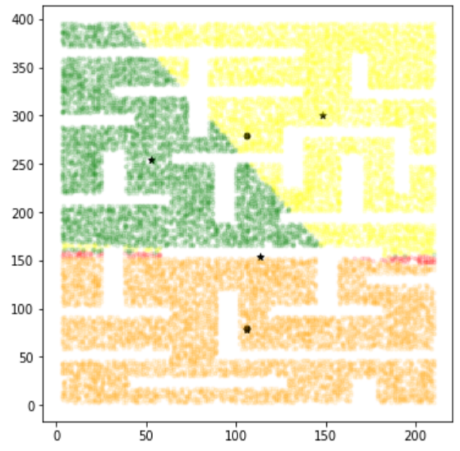

Figure 7: 4-cluster clustering at epoch 490 without ℓ_{inter}, combined.

5 Discussion and Next steps

In this work, we demonstrated k-means clustering using GraphSAGE model on a maze grid data. With loss function engineering, the GNN model is able to demonstrate k-means algorithm on the grid data. We found it is more difficult to apply clustering using a GNN on grids than real-world graphs such as citation networks or social media where there are naturally occurring community structure. In maze grids, most nodes are uniformly connected ($d = 4$), and they do not have naturally occurring communities or clusters. We used Euclidean loss functions with a few regularization terms and demonstrated k-means clustering using GNN. The results show that it works with some limitations: (1) k-means clustering of nodes in a maze grid using Euclidean distance ignores walls and cluster the nodes that are on the other side of the wall. (2) The current model (2-layer GraphSAGE) only covers 2-hop neighborhood per node when making decisions, thus it uses very little neighborhood information. A bigger field of view might be beneficial. For that, a deeper architecture may help, but also it may cause over-parametrization. To overcome it, we can use simpler layers such as lightGCN and increase the number of layers in GNN. Another idea we propose is to augment the graph to have more clustered structure. If the graph has more clustered structures, it is easier to cluster using modularity information. Tsitsulin *et el.,* demonstrated clustering using modularity-based loss function on paper-citation networks and more [2].

Inspired by the approach in [2], we explored the idea of augmenting the raw grid to a more densely connected graph. We applied an edge augmentation to increase the number of edges between nodes that are nearby each other. To perform this edge augmentation, we applied a random walk from each node, and then added edges based on the number of times each neighboring node was visited. Adding these additional edges can help performance of the GNN to increase its field of view with small number of layers as well as it can allow us to use modularity-based loss for clustering.
Figure 10 shows an example of 5-hop-randomwalk. Figures 8 and 9 show connections in the raw grid and randomwalk-augrmented graph. Figure 11 shows a heat map of node degrees of the nodes in the augmented graphs. One can see that there are clustered structures. Both these clusters and the randomwalk edges can make the clustering algorithm more effective by modularity information and increased range for reaching neighborhood.

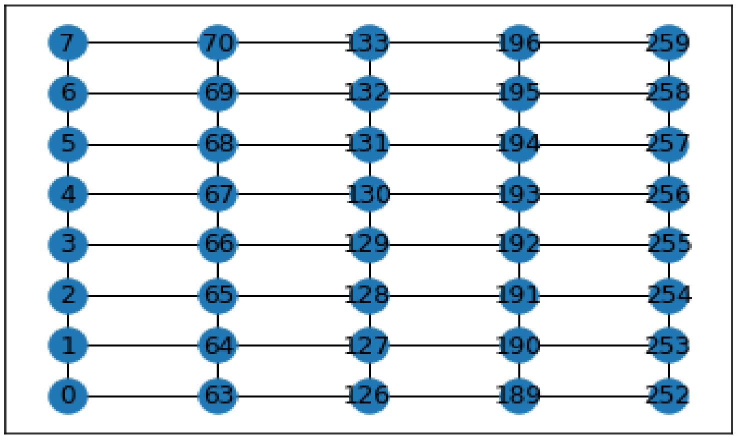

Figure 8: Subgraph generated based on coordinates adjacency

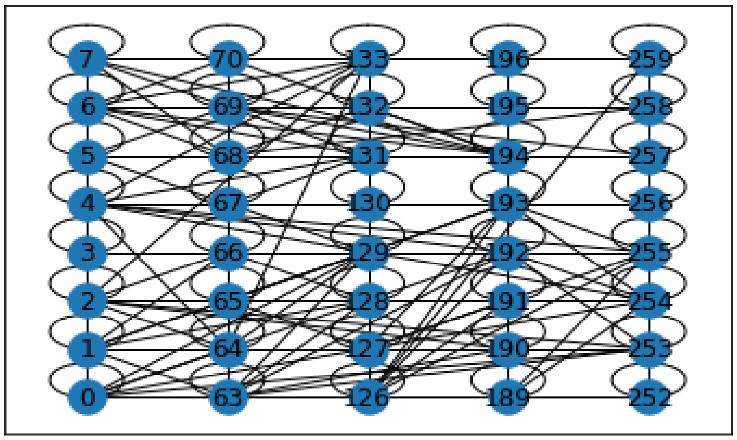

Figure 9: Augmented subgraph

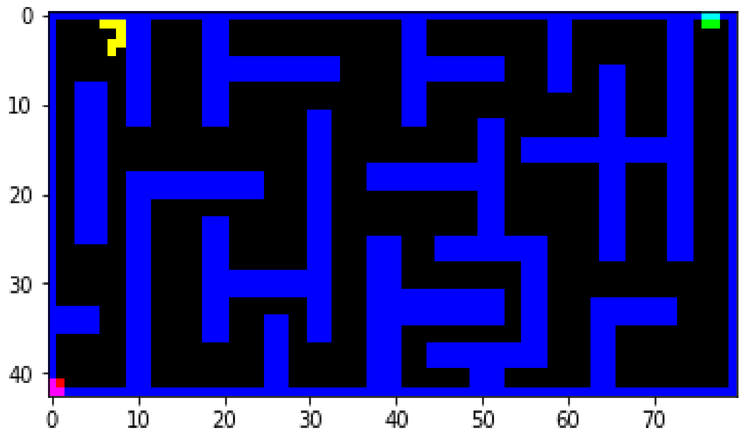

Figure 10: Random walk from a single node

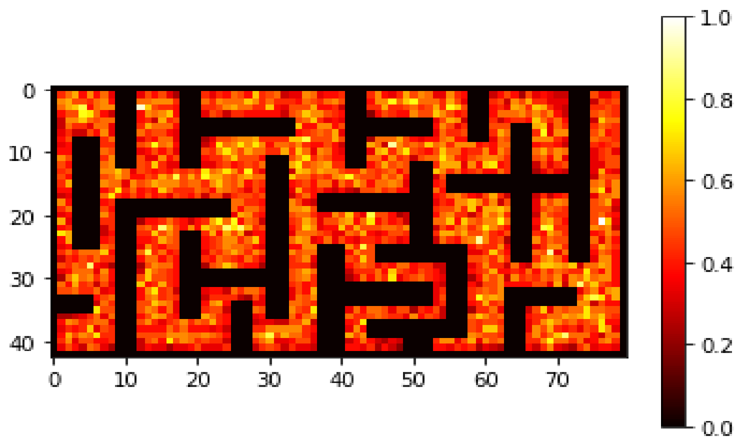

Figure 11: Heatmap of degree distribution after edge augmentation

References

[1] Hamilton, W.L., Ying, R., Leskovec, J.: Inductive Representation Learning on Large Graphs. arXiv:1706.02216 (2018).

[2] Tsitsulin, A., Palowitch, J., Perozzi, B., Müller, E.: Graph Clustering with Graph Neural Networks. arXiv:2006.16904. (2020).