CS230

Multiclass Graph Neural Network for Single Cell RNA sequencing Classification

Hongyi Ren
Department of Electrical Engineering
Stanford University
hongyi@stanford.edu

1 Introduction

In many biomedical experiments, how to identify and characterize the cellular composition of complex tissues becomes a key step. Single-cell RNA sequencing (scRNA-seq) provides unprecedented opportunities to complete these tasks. Over the past decade, technological advances have allowed scRNA-seq technologies to scale to thousands of cells per experiment. [14] A common analysis step in analyzing single-cell data involves the identification of cell populations presented in a given dataset. This task is typically solved by unsupervised clustering of cells into groups based on the similarity of their gene expression profiles, followed by cell population annotation by assigning labels to each cluster. However, the annotation step is cumbersome and time-consuming as it involves manual inspection of cluster-specific marker genes. Moreover, one most significant challenge is that there are huge amount of classes. Usually dataset like the Tabula Muris[2] have a long-tail problem with some minor classes having few numbers of training data. (See Figure 1b). In order to reduce the cost of manual inspection and the problem caused by few-shot classes, we propose a Graph Neural Network (GNN) model based episodic learning model. The input will be the scRNA-seq data, which is a 1089 dimension vector, and the output will be one of the 55 classes. By comparing the MLP model, GCN model and KNN models, we showed that our model can perform better in few-shot learning problems when the class numbers are extremely large.

2 Related work

In 2017, Zhengyan, Li, and Chi, [15] classifed lung adenocarcinoma and Squamous cell carcinoma using RNA-Seq Data. Tey used gene expression profle to discriminate NSCLC (Non-Small Cell Lung Cancer) patient's subtype. To construct classifers based on the training data, they considered methods include Logistic Regression with Principle Component Analysis(PCA), logistic regression with LASSO shrinkage (LASSO), and KNN. Besides the traditional methods, MLP is applied in Micheal O. Arowolo's survey [16] with the advance of deep learning during the most recent 15 years. They test it on the TCGA dataset and get better accuracies compared with the logistic regression and KNN methods.

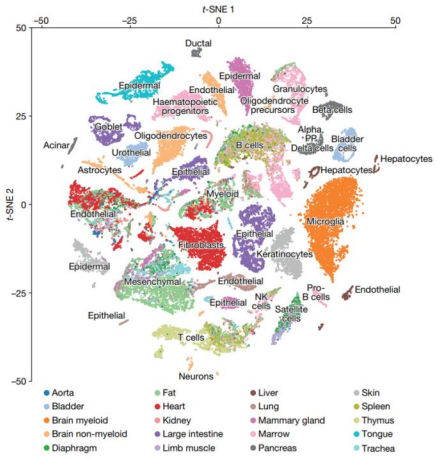
However, most of these results only report accuracy as metrics. Considering that the problem usually has a large amount of classes and a long tail of few-shot classes, the accuracy can't represent the actual evaluation of the classification models. Even the model performs well in the dominant classes, it might not as good as the accuracy shows for the few-shot classes.

To deal with this problem, the Prototypical Networks with episode learning is introduced[17]. The paper assumes that there exists a mathematical representation of the images, in which samples of the same class gather in groups called clusters. The main advantage of working in that embedding space is that two images that look the same will be close to each other, and two images that are

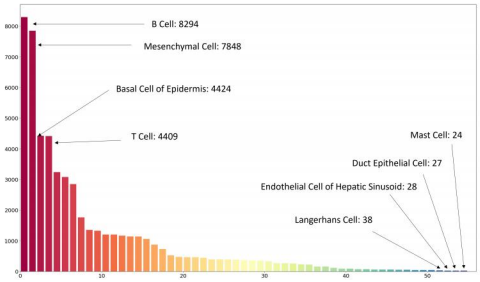
CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

completely different will be far away. In each training episode, some query data will be drawn with the neighboring samples and their labels as the support data. Then the network will project them in to certain embedding space. And in one training episode, the model will be trained to make the embedding of the query sample is close to embedding center of the support samples under the same classes.

With the innovation of the prototypical network, we proposed our GNN model with episode learning.



(a) t-SNE visualization of all FACS cells. t-SNE plot of all cells collected by FACS, coloured by organ, overlaid with the predominant cell type composing each cluster; n = 44,949 individual cells.



(b) The distribution of class samples. With the largest class having more than 8000 samples, the smallest class only has 24 samples. The colors and class names are in the appendix, Figure 5.

Figure 1: Datasets information.

3 Dataset and Features

The dataset is from the Tabula Muris scRNA-seq dataset[2]. The data allow for direct and controlled comparison of gene expression in cell types shared between tissues, such as immune cells from distinct anatomical locations. It contains 54865 samples in total. Each sample has 1089 features representing the dene expression. It has cells of 20 organs from four male and three female mice, which in total has 55 classes. Figure 1a shows a visualization of the t-SNE of the dataset from the oringinal paper, and Figure 5 shows the t-SNE visualization that we generated.

As we described in the Introduction, this dataset has a long tail problem, as Figure 1b shows. The largest class has 8294 samples, and the smallest class has 24 samples. In order to have training data for each class, we randomly sampled 20% in each class as testing data.

4 Methods

The basic pipeline of our model is as the Figure 2. In one episode, We firstly initialize the graph with the original features. When we construct the graph, we computed the euclidean distance between every pair of samples, and created an edge between a node and its top k nearest neighbors. One of the graphs is visualized in Figure 6 in Appendix. After we construct the graph, we feed it into a GCN network. We then optimize the GCN network based on the cross entropy loss of the classification task.

Once the training of the episode is done, we then collect the embedding of the last layer and create a new connected graph based on that. We then update the connected edges and weights of the edges, and train the next episode. By doing this episodically, we are updating our graphs so that nodes in the similar class will be connected eventually.

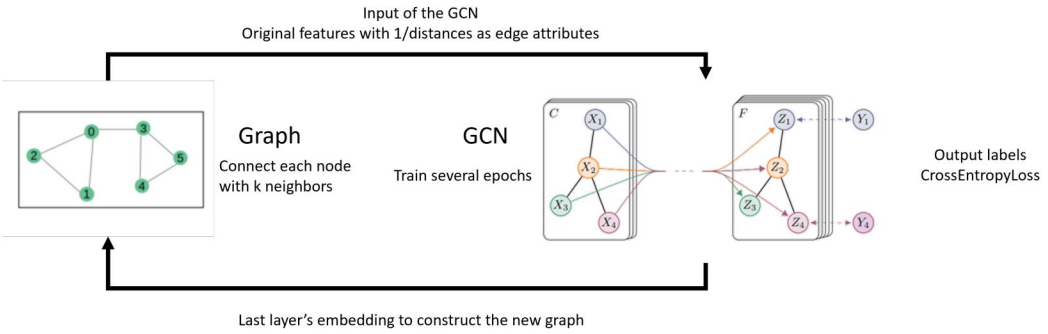
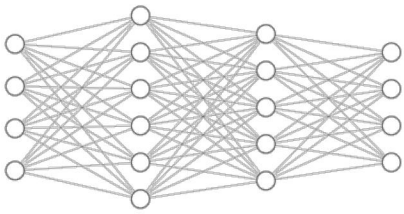
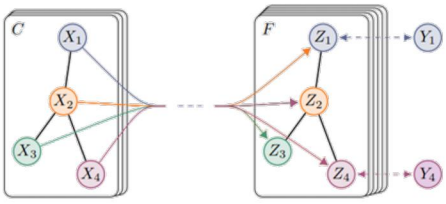


Figure 2: The training procedure of our GCN model. During every episode, we firstly construct the graph based on nodes' neighbors, annd then feed the graph into the GCN network. The GCN network will be trained for several epochs. Because the GCN network combines the embedding of each nodes' neighbors, even without much training samples, the model can still learn from the nodes' neighbors feature.



(a) We used a simple MLP which take all 1089 features as input, and two hidden layers with neurons 256 and 64. Each fully connected layers is followed by a ReLU activation and a Dropout with 0.5 keep prob. The output is a 55 dimension with softmax activation.



(b) In order to compare with MLP, the GCN has a very simialr architecture. It take all 1089 features as input. Each node in the hidden layers has 256 and 64 features.The activation is ReLU and the keep prob of Dropout is also 0.5. The output is a 55 dimension with softmax activation.

Figure 3: MLP and GCN models. Both of them have 299063 trainable parameters.

4.1 GCN and MLP network

For the purpose of showing MLP models can't handle the few-shot cases, we firstly trained a MLP model. In fact, it is also a special case of our graph construction when k=0. It only considers the sample itself when makes classifications. If there is no enough data in the training set, it can't have a good classifier. As the Figure 3a shows, the network has in total $(1089 \times 256 + 256) + (256 \times 64 + 64) + (64 \times 55 + 55) = 299063$ trainable parameters.

We then generate a graph by embedding the original data into a geometry preserving space [4] and connecting each sample with its k-nearest-neighbors(KNN)[3]. Similarly as learning a prototypical embedding in [11-13], the graph structure makes use of the neighbors of a sample, and therefore, with the samples from a minor class, learning from the graph structure can perform better than only considering the single node. Therefore GCN can have a better performance compared with the traditional method and a pure MLP network, especially for the minor classes.

After creating the graph, we applied the Graph Convolutional Network (GCN)[6] as the backbone of our model.The graph convolution layer can be written in this format:

$$\mathbf{x}'_i = \mathbf{\Theta}^\top \sum_{j \in \mathcal{N}(v) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{x}_j$$

with $\hat{d}_i = 1 + \sum_{j \in \mathcal{N}(i)} e_{j,i}$, where $e_{j,i}$ denotes the edge weight from source node j to target node i

$\mathbf{\Theta}^\top$ is the trainable weight, and it has the dimension of $\dim(\mathbf{x}') \times \dim(\mathbf{x})$. So we set the features of nodes in each layer equal to the hidden neurons in MLP to keep these two models has same number of trainable parameters.

5 Experiments/Results/Discussion

For the implementation, we use PyTorch [9] and PyTorch Geometric (PyG) [10]. Most of the operations in GNN are implemented in the PyG package. For the experiments, we have trained both the MLP and GCN for 200 Epochs in total with Adam Optimizer. With GCN network, we actually trained 5 episodes with 40 epochs for each episode.

We show the results in the Figure 4.

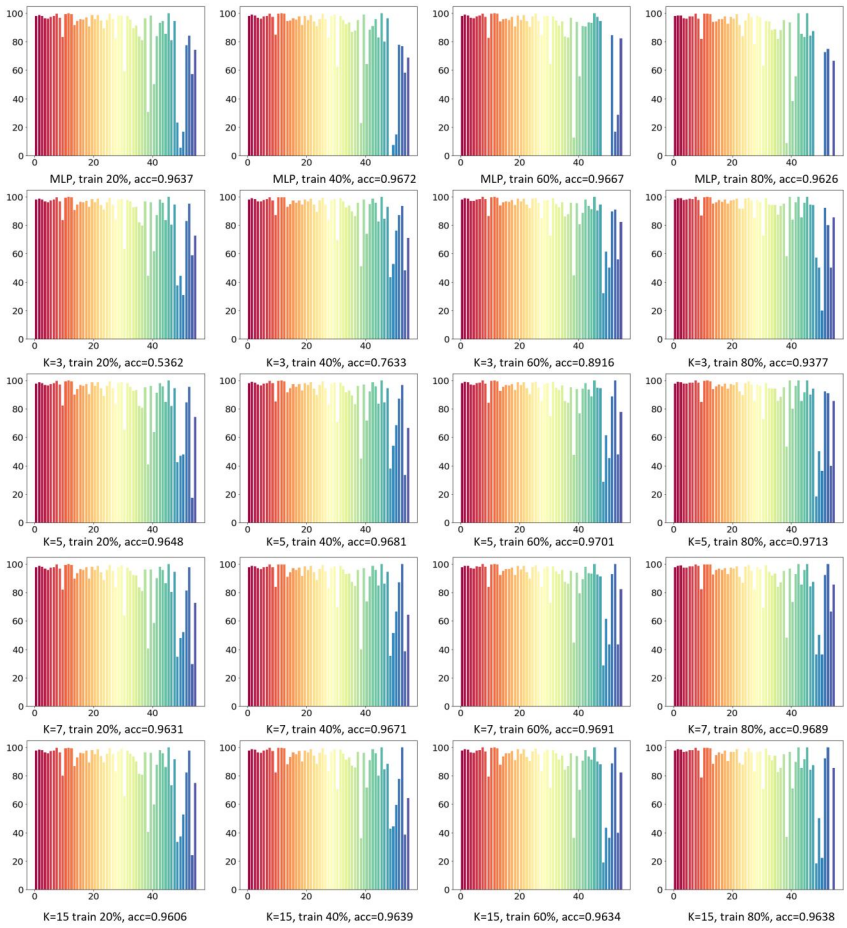


Figure 4: Testing F1 scores of different classes. Bar colors are the same as Figure 1b. The left bar represents the largest class. The first row shows the results from the MLP model (k=0). K means the number of neighbors that the graph connect. And in order to show that the GCN performs well in cases with even less samples, we also added experiments with less training data (percentage shows the training data versus the total number in the whole dataset).

With 80% data as trainingi data, The KNN model with 7 neighbors has 94.59% average accuracy on the test dataset. With the power of deep learning, the MLP model has 96.26% average accuracy and the GCN model with 7 neighbors-connected graph has 96.89% accuracy. However, although the MLP model has higher accuracy, from the F1 scores of each score in Figure **??**, we can see that it performs terrible in the minor classes, while KNN and GCN did better in those few-shot learning cases. And the table shows the average F1 scores for different experiments.

K	Avg F1 (20%)	Avg F1 (40%)	Avg F1(60%)	Avg F1(80%)
0 (MLP)	0.8643	0.8214	0.7826	0.7521
3	0.9368	0.9486	0.9684	0.9657
5	0.9465	0.9486	0.9594	0.9632
7	0.9541	0.9623	0.9680	0.9714
15	0.9657	0.9679	0.9564	0.9457

6 Conclusion/Future Work

From the results, our work shows that our model can handle the cases when there is not enough data. For the interpretability, we are innovated by the GNNexplainer [7] method. However, since it is designed for academic publication dataset, which is different from our case, we are planning to modify some part in the node feature explainer part. Our idea is to figure out the interaction between the gene expression features instead of merely showing which features are important.

7 Contributions

All of the coding work is done by Hongyi Ren. The training work is done on the servers and sponsored by Prof. Lei Xing, Radiation Oncology department, Stanford.

References

References

[1] Dries R, Zhu Q, Eng C, Sarkar A, Bao F, George R, et al. Giotto, a pipeline for integrative analysis and visualization of single-cell spatial transcriptomic data. bioRxiv. 2019;701680

[2] The Tabula Muris Consortium., Overall coordination., Logistical coordination. et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature 562, 367–372 (2018).

[3] Fix, Evelyn, and Joseph Lawson Hodges. "Discriminatory analysis. Nonparametric discrimination: Consistency properties." International Statistical Review/Revue Internationale de Statistique 57.3 (1989): 238-247.

[4] Islam, Md Tauhidul, and Lei Xing. "Geometry and statistics-preserving manifold embedding for nonlinear dimensionality reduction." Pattern Recognition Letters 151 (2021): 155-162.

[5] Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2009.

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

[7] Ying, Zhitao, et al. "Gnnexplainer: Generating explanations for graph neural networks." Advances in neural information processing systems 32 (2019).

[8] Yuan Y, Bar-Joseph Z. GCNG: graph convolutional networks for inferring gene interaction from spatial transcriptomics data. Genome Biol. 2020 Dec 10;21(1):300. doi: 10.1186/s13059-020-02214-w. PMID: 33303016; PMCID: PMC7726911.

[9] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems 32 (2019).

[10] Fey, Matthias, and Jan Eric Lenssen. "Fast graph representation learning with PyTorch Geometric." arXiv preprint arXiv:1903.02428 (2019).

[11] Ding, Kaize, et al. "Graph prototypical networks for few-shot learning on attributed networks." Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020.

[12] Prabhu, Viraj, et al. "Prototypical clustering networks for dermatological disease diagnosis." arXiv preprint arXiv:1811.03066 (2018).

[13] Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." Advances in neural information processing systems 30 (2017).

[14] Svensson, Valentine, Roser Vento-Tormo, and Sarah A. Teichmann. "Exponential scaling of single-cell RNA-seq in the past decade." Nature protocols 13.4 (2018): 599-604.

[15] Huang, Z., L. Chen, and C. Wang. "Classifying lung adenocarcinoma and squamous cell carcinoma using RNA-Seq data." Cancer Stud Mol Med Open Journal 3.2 (2017): 27-31.

[16] Arowolo, Micheal Olaolu, et al. "A survey of dimension reduction and classification methods for RNA-Seq data on malaria vector." Journal of Big Data 8.1 (2021): 1-17.

[17] Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." Advances in neural information processing systems 30 (2017).

Appendix

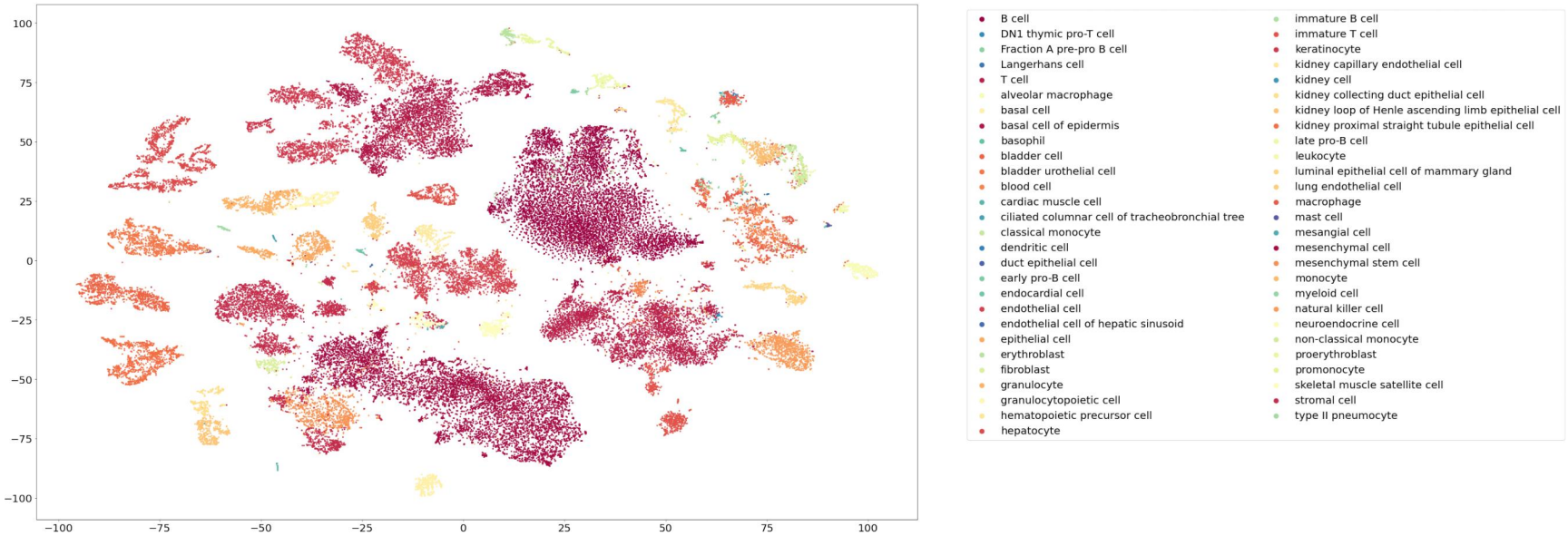


Figure 5: The t-SNE visualization of the dataset. The colors are same as Figure 1b, with red corresponding to major class and blue corresponding to minor class.

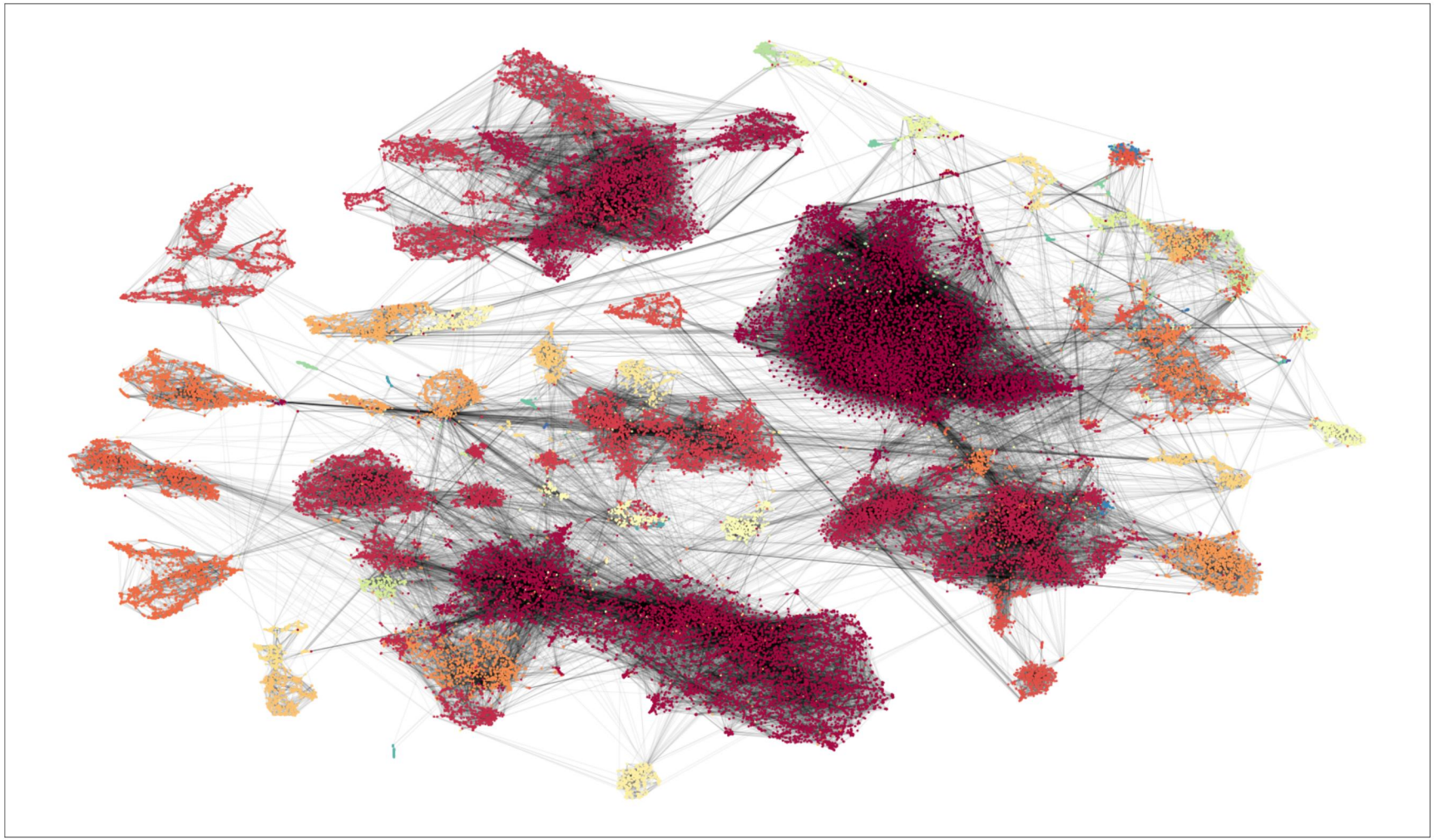


Figure 6: The visualization of the graph. We connect each node with 7 neighbors. The distance metrics is euclidean distance.