DevoGraph: Modeling Cell Developmental Process of C. elegans using Graph Neural Network

GSoC 2022 Project Idea 22.1: GNNs as Developmental Networks

Contack details

Name: Jiahang Li
Email: lspongebobjh@gmail.com; ljh1064126026@bupt.edu.cn
Slack: lspongebobjh@gmail.com
Location: Beijing, China (GMT+8)
Github: https://github.com/LspongebobJH
Education: Senior student, Data Science and Big Data Technique affiliated with Computer Science Department, Beijing University of Posts and Telecommunications
CV: https://github.com/LspongebobJH/me/blob/main/Jiahang_Li.pdf

Project abstract

Biological development features many different types of networks: neural connectomes, gene regulatory networks, interactome networks, and anatomical networks. Using cell tracking and high-resolution microscopy, we can reconstruct the origins of these networks in the early embryo of *C. elegans*[1].

Specifically, we'd like to model cells and early embryos as different kinds of graphs given microscopy videos of these objects. The node and edge features will be built based on biological properties and observation on microscopy data. Classic Graph Neural Networks (GNN), such as GraphSAGE[2], GCN[3], GAT[4], will be employed on these graphs to obtain high-level embeddings from input features for each node and the whole graph. The embeddings can be utilized in downstream tasks.

Our objective is to build an open-source framework called DevoGraph, which models cells' developmental process as graphs, extracts and converts latent information into high-level embedding using tools of GNN. The embeddings can be utilized in a series of downstream tasks regarding *C. elegans* from the perspective of network[1][5][6], such as analyzing cell tracking, cell division and differentiation, lineage trees, cell

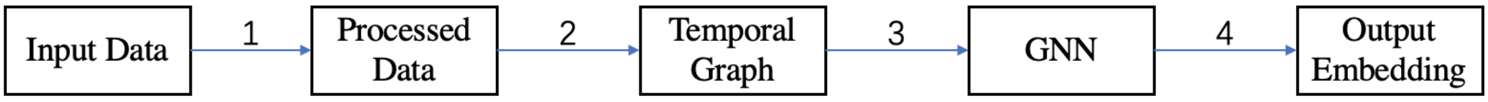
network structures in embryogenesis, cell cluster, cell populations, developmental process of *C. elegans*, etc.

The project will be quite beneficial and helpful for the mentoring organizations INCF[7], OpenWorm[8] and the mentoring subgroup DevoWorm[9]. INCF is an international organization that promotes sharing and integrating neuroscience data and knowledge worldwide. OpenWorm is an open-source project that is dedicated to building the first comprehensive computational model of the *C. elegans.* DevoWorm is engaged in biological data analysis by using tools of machine/deep learning, etc. Our projects are aligned with the research directions of these organizations. DevoLearn focuses on neuroscience and will provide a series of deep learning tools that help mentoring organizations, groups and other users utilize GNN to analyze *C. elegans* and its network structure. And follow-up research works regarding the intersection of GNN, network analysis and neuroscience will benefit a lot from DevoGraph. We also hope that DevoGraph will motivate more open-source projects that introduce machine/deep learning techniques into neuroscience.

Project details

The project can be decomposed into several steps:

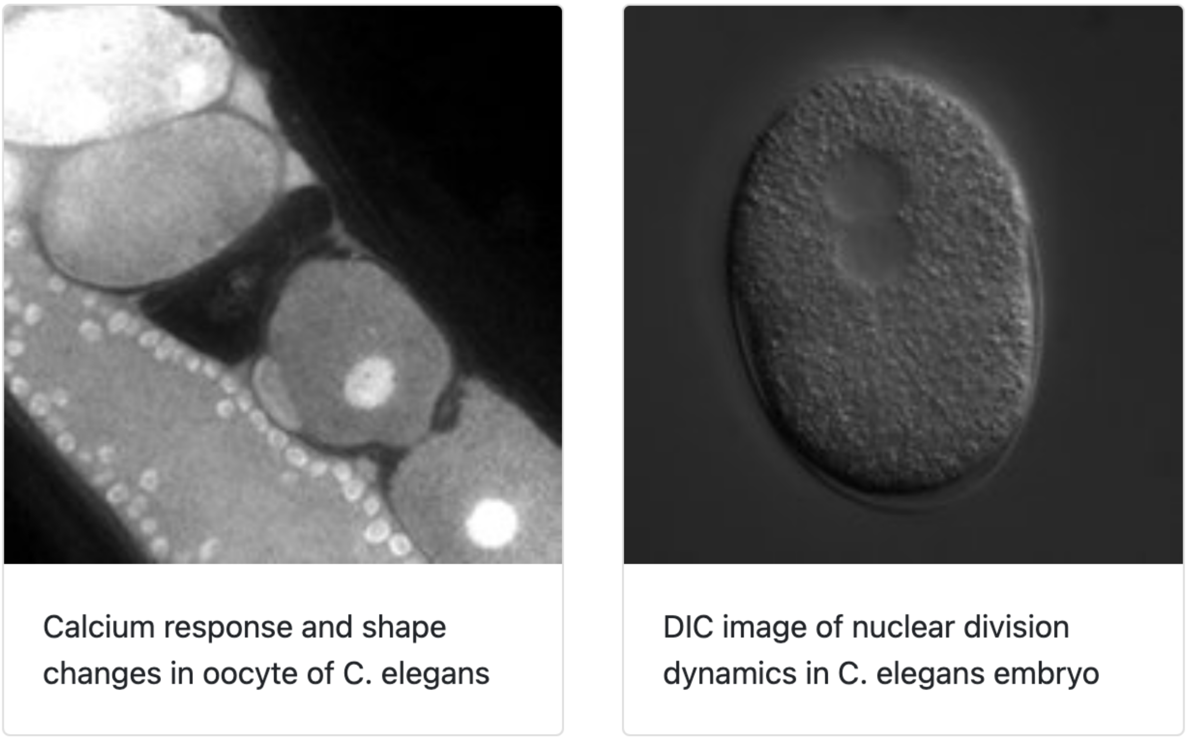
1. Convert input data into processed data. The input data include raw microscopy images and videos of *C. elegans*, which can not be directly converted into graph structures. Here we utilize DevoLearn and other frameworks of computer vision to extract useful information as processed data from input data, which will be used to construct graphs.
2. Convert processed data into temporal graphs. DevoGraph models cell networks as temporal graphs which will dynamically change over time. This type of graphs can obtain the sequence information of time-lapsed images as well as network information, which is better than modeling embryos as static graphs.
3. Employ classic Graph Neural Networks (GNN) with the support of Recurrent Neural Network (RNN) on generated temporal graphs.
4. Train the GNN models and obtain high-level embeddings of nodes and graphs.



1. Convert input data into processed data

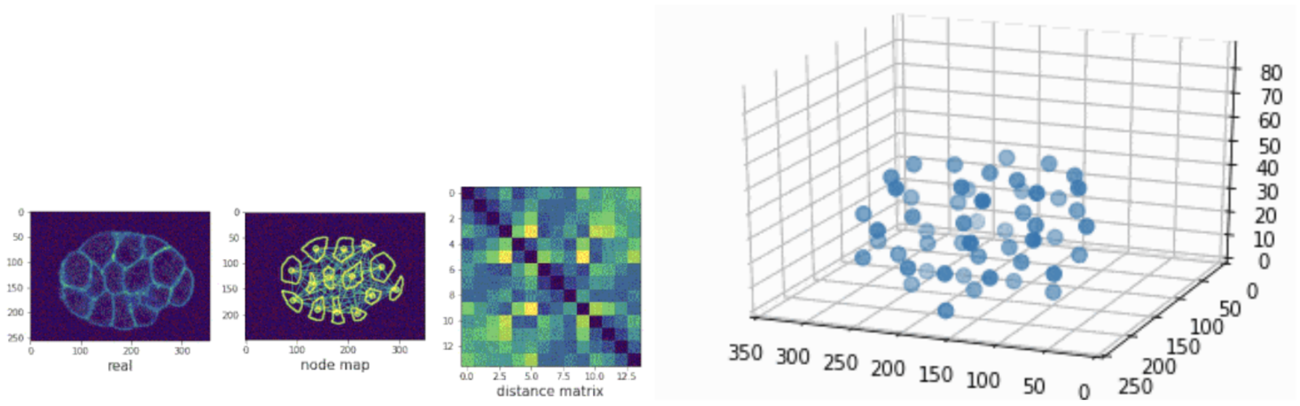
Our project is based on microscopy images and videos. Before constructing the network of cells and early embryos, we need to extract useful information from raw images that can be utilized to build graphs.

The raw microscopy datasets include: a large database for biological dynamics which provides a rich set of open resources for analyzing quantitative data and microscopy images of biological objects called SSBD[11], a dataset of DevoWorm team called DevoZoo[16], and some datasets regarding tasks on *C. elegans*, such as cell segmentation, etc.[12][13][14]

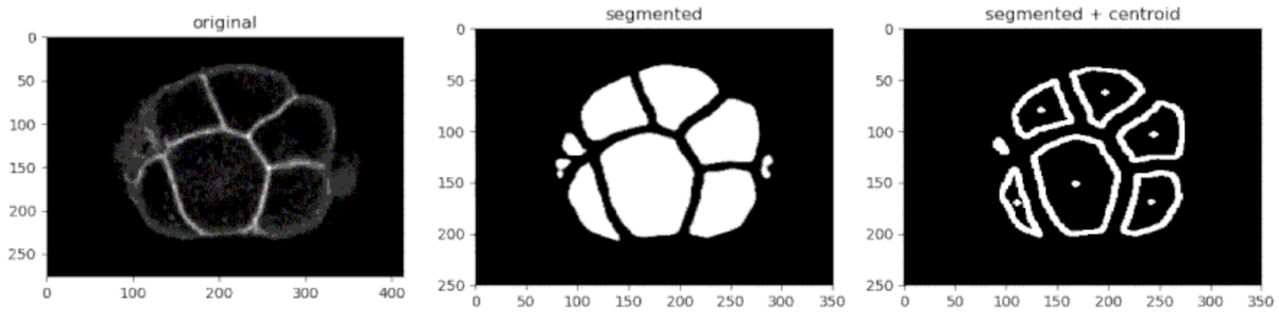

Calcium response and shape changes in oocyte of C. elegans
DIC image of nuclear division dynamics in C. elegans embryo

Examples of raw microscopy data of C. elegans from SSBD[11]

With the support of classic Convolutional Neural Network (CNN), such as ResNet[17], VGG[18], their pre-trained models, and other open-source deep learning frameworks for Computer Vision[19], our team has built DevoLearn[15], which is a toolkit for analyzing microscopy videos of cells and embryos using a CNN. DevoLearn can help us segment cells in *C. elegans* embryo, extract 2D or 3D centroids and generate synthetic images of embryos with a pre-trained GAN, etc.



Extract 2D and 3D centroids and compute Euclidean distance between centroids[10]



Segment cells separated by cell membrane[10]

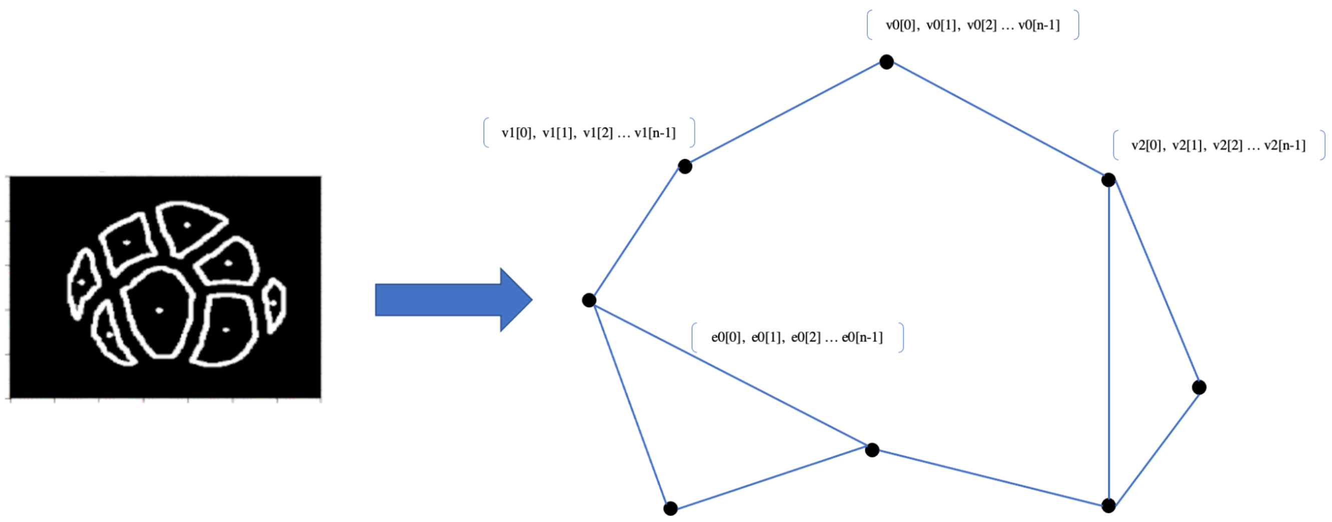
The output of DevoLearn, such as segmentation of cells, coordinates of centroids, distance between centroids, etc., is helpful for building graph structure, which is our next step.

Output

The output of this step includes some easy-to-use API that runs the pipeline of extracting information and building a processed dataset that stores the information. The format of the processed dataset is suitable for subsequent work based on the dataset. And the processed dataset will be provided as well.

2. Convert processed data into temporal(dynamic) graphs

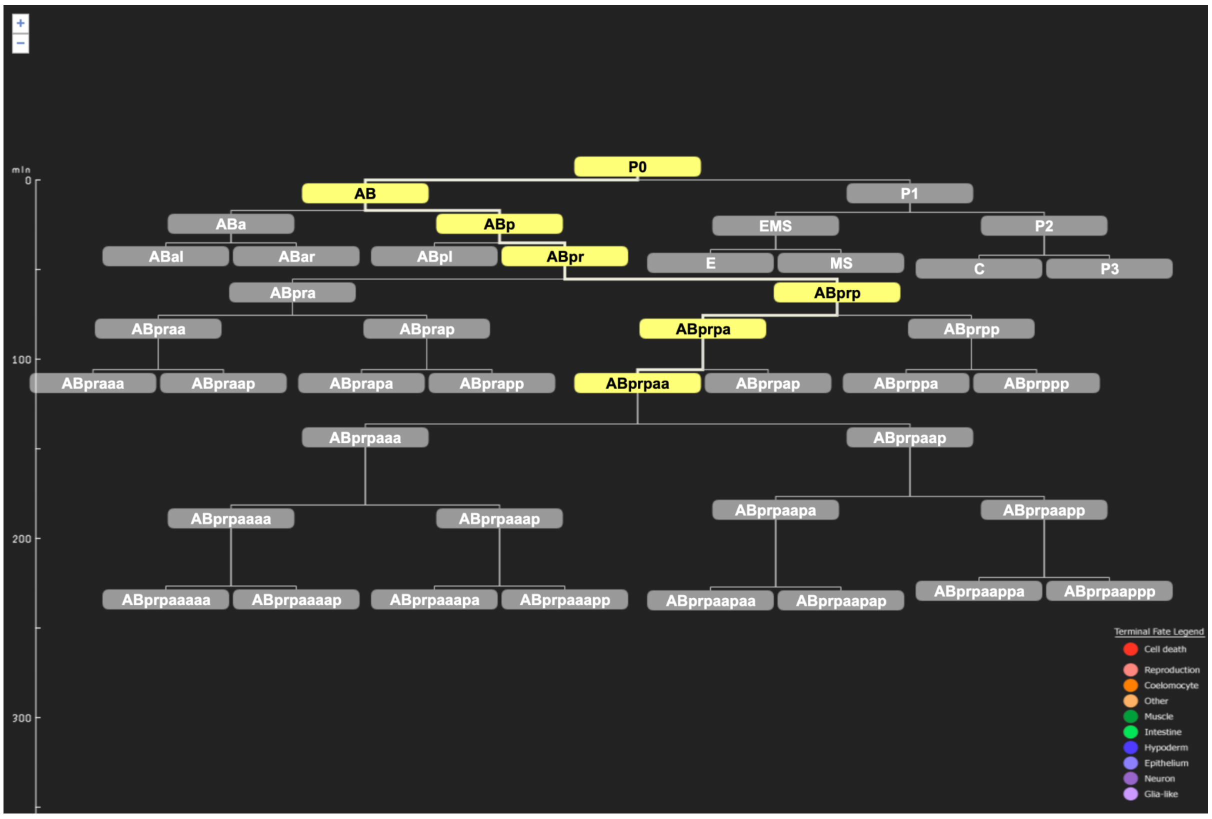
Inspired by TGN[23], multi-target tracking[24], cell tracking using GNN[5], etc., we design the temporal graph structure as follows.


v0[0], v0[1], v0[2] … v0[n-1]
v1[0], v1[1], v1[2] … v1[n-1]
v2[0], v2[1], v2[2] … v2[n-1]
e0[0], e0[1], e0[2] … e0[n-1]

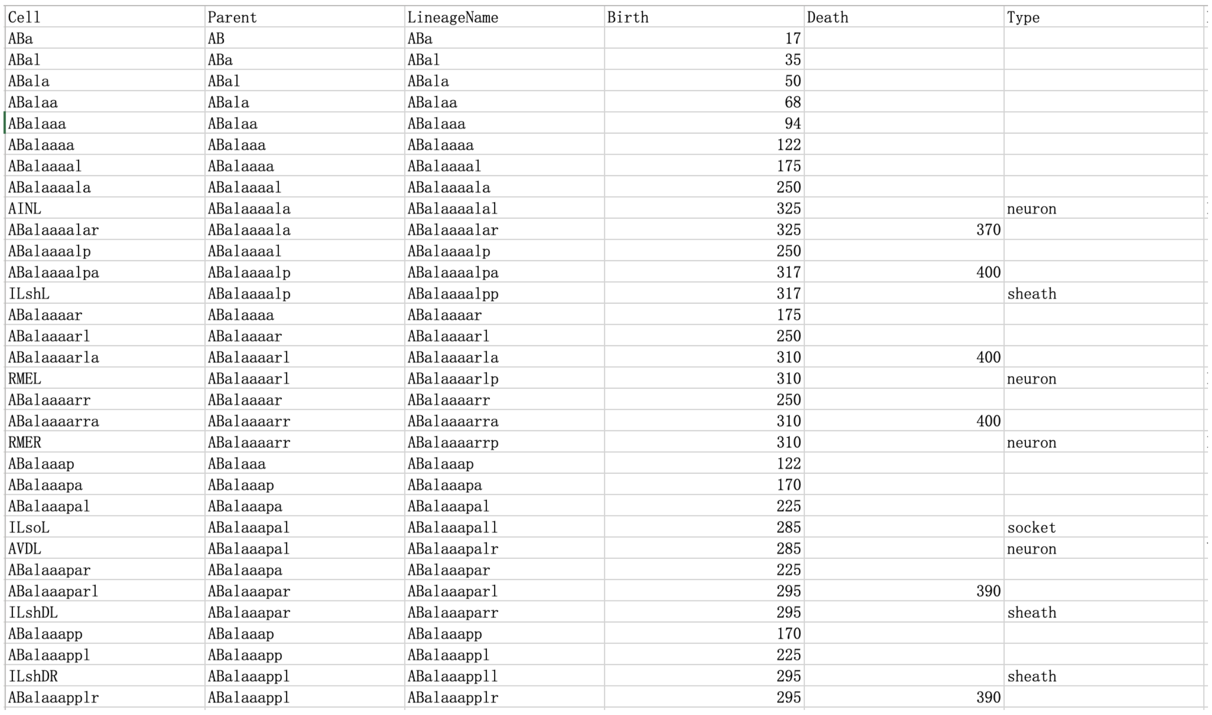
A simple example of graph constructed from a C. elegans embryo image

Each node in the graph represents the centroid of a cell. The interaction between cells is modeled as edges based on multiple rules. For example, we can set up a threshold value that only the node pairs with distance less than the threshold will be connected by an edge.

Node and edge features are also important. Observations on images including coordinates of centroids, distance between nodes, etc., and biological properties of cells including the cell lineage tree of *C. elegans*, birth and death time of cells, etc., are utilized to construct node and edge input features. The observations can be obtained from the output of the last step, while the biological information comes from prior biological knowledge and other datasets.



A part of C. elegans lineage tree[15]



Cell	Parent	LineageName	Birth	Death	Type
ABa	AB	ABa	17		
ABal	ABa	ABal	35		
ABala	ABal	ABala	50		
ABalaa	ABala	ABalaa	68		
ABalaaa	ABalaa	ABalaaa	94		
ABalaaaa	ABalaaa	ABalaaaa	122		
ABalaaaal	ABalaaaa	ABalaaaal	175		
ABalaaaala	ABalaaaal	ABalaaaala	250		
AINL	ABalaaaala	ABalaaaalal	325		neuron
ABalaaaalar	ABalaaaala	ABalaaaalar	325	370	
ABalaaaalp	ABalaaaal	ABalaaaalp	250		
ABalaaaalpa	ABalaaaalp	ABalaaaalpa	317	400	
ILshL	ABalaaaalp	ABalaaaalpp	317		sheath
ABalaaaar	ABalaaaa	ABalaaaar	175		
ABalaaaarl	ABalaaaar	ABalaaaarl	250		
ABalaaaarla	ABalaaaarl	ABalaaaarla	310	400	
RMEL	ABalaaaarl	ABalaaaarlp	310		neuron
ABalaaaarr	ABalaaaar	ABalaaaarr	250		
ABalaaaarra	ABalaaaarr	ABalaaaarra	310	400	
RMER	ABalaaaarr	ABalaaaarrp	310		neuron
ABalaaap	ABalaaa	ABalaaap	122		
ABalaaapa	ABalaaap	ABalaaapa	170		
ABalaaapal	ABalaaapa	ABalaaapal	225		
ILsoL	ABalaaapal	ABalaaapall	285		socket
AVDL	ABalaaapal	ABalaaapalr	285		neuron
ABalaaapar	ABalaaapa	ABalaaapar	225		
ABalaaaparl	ABalaaapar	ABalaaaparl	295	390	
ILshDL	ABalaaapar	ABalaaaparr	295		sheath
ABalaaapp	ABalaaap	ABalaaapp	170		
ABalaaappl	ABalaaapp	ABalaaappl	225		
ILshDR	ABalaaappl	ABalaaappll	295		sheath
ABalaaapplr	ABalaaappl	ABalaaapplr	295	390	

Lineage tree in the form of table which shows birth and death time of each cell and mature cell type[16]

As for node features, we will concatenate different information represented by different vectors as the node features. Specifically, node *v1* has 2D/3D coordinates, birth time, death time, the timestamp of the snapshot in the video, and other information. This information will be encoded as multiple vectors and be concatenated together as a new *n-dimensional* vector. The same constructing process can be applied to edge features.

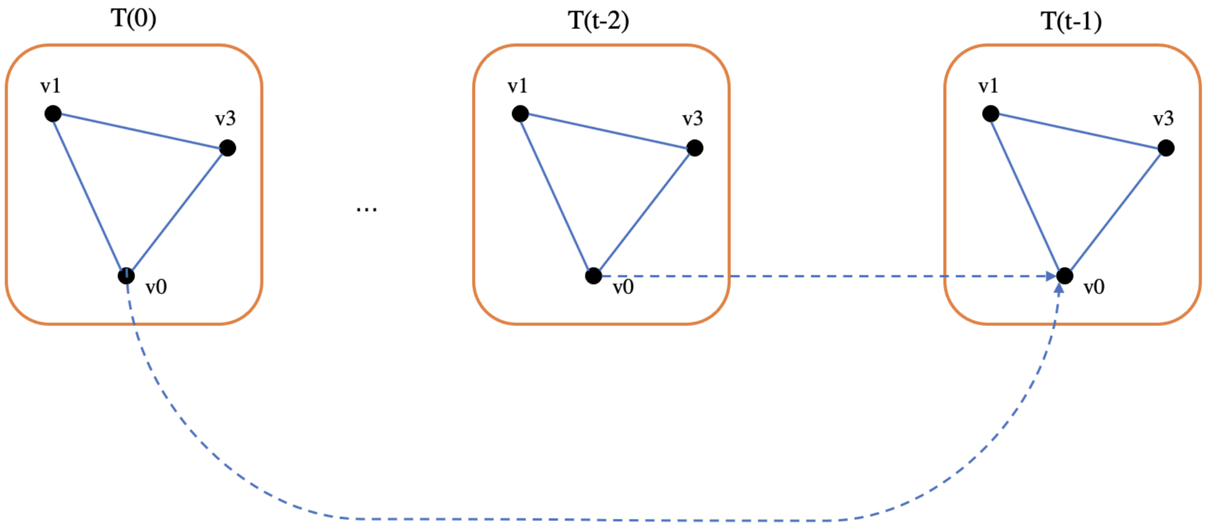
Note that the graph is dynamic. Its structure and features will change over time, that is, the image(snapshot) at each discrete timestamp will be represented as a unique graph. Some rules regarding how to deal with the appearance and disappearance of nodes and edges will be set up according to research works on dynamic graphs and *C. elegans*. For example, cells not present in the previous image will have an all-zero feature vector[23]. In DevoGraph new cells will be initialized based on their biological properties and other information instead of a simple initialization as the example.

Output

The output of this step includes some easy-to-use API that generates a series of graphs with node and edge features based on the processed dataset given by step 1. And the generated graphs will be provided as well. The format of graphs will probably be of DGL[20], etc, which are commonly-used graph learning frameworks. This setting will help us implement GNN on graphs conveniently.

3. GNN being employed on graphs

Our project objective is to model the developmental process of cells, that is, simply employing existing GNN on graphs is not enough. Because most existing GNN are designed for *static graphs*, while our graphs in the project are *temporal graphs* over videos. Inspired by classic works in temporal network embedding and multi-target tracking, etc., we'd like to combine RNN[25] and its variants (LSTM[26], GRU[27], etc) with GNN to obtain sequence and graph information simultaneously.



A simple example of how to combine RNN and GNN on a temporal graph

Each snapshot modeled as a graph can be regarded as an input at each time step. For example, we firstly employ GNN on snapshot of *T(t-1)* and obtain:

$$m(v_0|T(t-1)) = act(MSG(v_0, N(v_0)))|T(t-1)$$

MSG is the message propagation defined by the specific GNN layer, such as GCN, GraphSAGE, GAT, etc. *act* is the activation function, such as ReLU[29], Sigmoid[30], Tanh[31], etc. *N(v0)* contains features of neighbors of *v0* and edges that connect to *v0*. The *v0* here represents the feature vector of itself.

We can regard *m(v0 | T(t-1))* as the hidden state at the time step *T(t-1)*. Then we aggregate hidden states from preceding time steps into *T(t-1)*.

$$h(v0|T(t-1)) = SEQ(m(v0|T(0)), m(v0|T(1)), \ldots, m(v0|T(t-1)))$$

SEQ is the aggregation function along the sequence, which can be RNN and its variants, attention mechanism and its variants, etc.

Output

The output of this step includes implementations of some classic GNN for constructed graphs in step 3. And the models will be based on DGL[20], etc., for simpler implementations. The pre-trained models will be provided as well if there's no existing ones.

4. Output embedding

By now, *h(v0 | T(t-1))* is the output embedding that has extracted latent information from the whole sequence and graph and can be used in downstream tasks for C. elegans analysis, such as cell tracking, cell movements prediction, cell parent prediction, etc. Moreover, we can employ a pooling function on *h(g | T(t-1)),* where v include all nodes in the graph, to obtain a high-level representation of the graph.

Follow-up works

If time permits, we'll design some tasks regarding the developmental process of *C. elegans*, such as cell tracking, etc., and utilize our project to analyze these tasks.

Timeline

May 20 - June 12

Community Bonding Period.

- Interact with mentors and other teammates to get familiar with the community.
- Check related papers, datasets and frameworks which can support our project.
- Clear out all uncertainties regarding the project by communicating with the mentors.

June 13

Start coding

Week 1 - Week 3

June 13 - July 3
Work on Step 1: Convert input data into processed data.

- Check classic research papers related to C. elegans so that we can understand what information is important when analyzing the developmental process of C. elegans.
- Check details of C. elegans-related datasets, and figure out how to utilize DevoLearn, torchvision, etc., to extract information from these images and videos.
- Provide the API that runs the pipeline of extracting information and building a processed dataset that stores the information.
- Provide the processed datasets.
- Complete documents, unit tests and examples of how to use datasets.

Week 4 - Week 6

July 4 - July 24
Work on Step 2: Convert processed data into temporal(dynamic) graphs

- Check research papers related to temporal(dynamic) graphs and GNN that work on them to figure out how to effectively model C. elegans early embryos as graph structures..
- Check biological information of C. elegans that help with building expressive node and edge features.
- Provide API that generates a series of graphs with node and edge features.
- Provide generated temporal graphs.
- Complete documents, unit tests and examples of how to use graphs.

Week 7

July 24 - July 29
Phase 1 Evaluation

Week 8 - Week 10

August 1 - August 21
Work on Step 3 and Step 4: Employing GNN on graphs to obtain high-level embedding

- Check research papers related to (temporal) graph neural network, recurrent neural network, multi-target tracking, (multivariate) time-series analysis, cell tracking, C. elegans (developmental process) analysis, etc. These works will help us design and develop models for C. elegans.
- Provide API and implementations that utilize DGL or other frameworks to implement classic GNN models for constructed graphs.
- Provide pre-trained models for analyzing C. elegans if there's no existing ones.
- Complete documents, unit tests and examples of how to API and pre-trained models.

Week 11 - Week 12

August 22 - September 4

- Refine the previous code implementation, tests, and documents and fix bugs.
- Utilize implemented models and generated high-level embedding on downstream tasks if time permits. The tasks include cell tracking, cell movements prediction, etc.

Last week

September 5 - September 12

Final evaluation.

- The project will provide multiple API with documents about how to use them. These API include processing data from raw microscopy data, constructing graphs from processed data, training models on graphs and obtaining embeddings by inferencing the pre-trained models.
- The project will provide python scripts that contain examples and pipelines of using the API. The shell scripts that run the python files and output some visualization and analysis results will also be provided.
- The project will provide Jupyter notebooks that contain how to visualize the embeddings by mapping them into low-dimensional vectors, how to analyze *C. elegans* based on project, how to employ our project on some basic tasks related to *C. elegans*, etc.

Future directions

I have some research ideas based on our project, which is related to cell movement prediction, seq2seq and graph generation. I'll discuss it with my mentors.

Other plans

I'll keep mentors updated, discuss details of work and what is ahead of us twice a week by Slack, weekly video calls and emails. This project is the only one that I apply for. Specifically, I'll spend 15-20 hours per week. And I'm preparing my dissertation at the same time. But I will coordinate my work on DevoGraph accordingly so that they will not conflict with each other.

About me

My research interests cover graph (neural network) representation learning and its application, large-scale distributed systems for graph analysis, etc. I have some related open-source frameworks contribution experience.

For example, I've been a software development engineer intern from July 2021 (until June this year) in Amazon Shanghai AI Lab[21], which is directed by Prof. Zheng Zhang. My main works include contributing to DGL[20], an open-source framework for deep learning on graphs, and working on research projects regarding graph neural networks with Dr. David Wipf and Dr. Xiang Song. We have one paper regarding GNN on arxiv, called Network in Graph Neural Network[28].

And I’ve worked as a research assistant directed by Prof. Chuan Shi from Feb 2021 to July 2021 in GAMMA Lab affiliated with Beijing University of Posts and Telecommunications. My main works include maintaining and contributing to OpenHGNN[22], an open-source framework that eases research work of heterogeneous graph neural networks. And I’m one of the earliest contributors of OpenHGNN. Please refer to my CV for more details.

I have some understanding of graph neural networks (GNN) and I’m quite interested in its applications on biological analysis. The project is to employ GNN on a biological topic, that is, the developmental process analysis of *C. elegans*. That’s why I’m interested in participating in the project. Moreover, my experience and knowledge regarding GNN and open-source softwares contributions are aligned with the requirements and target of the project, which will be helpful for building the DevoGraph.

In conclusion, I wish I can be admitted as a contributor to this project. And I believe I will finish it well.

Appendix

[1] Alicea, B., & Gordon, R. (2018). Cell Differentiation Processes as Spatial Networks: identifying four-dimensional structure in embryogenesis. BioSystems, 173, 235-246.
[2] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.
[3] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
[4] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.
[5] Ben-Haim, T., & Riklin-Raviv, T. (2022). Graph Neural Network for Cell Tracking in Microscopy Videos. arXiv preprint arXiv:2202.04731.
[6] Wang, P. Y., Sapra, S., George, V. K., & Silva, G. A. (2021). Generalizable machine learning in neuroscience using graph neural networks. Frontiers in artificial intelligence, 4.
[7] INCF: A standards organization for open and FAIR neuroscience. https://www.incf.org/
[8] OpenWorm: An open source project dedicated to creating the first virtual organism in a computer. https://openworm.org/
[9] DevoWorm: Developing the Nematode (C. elegans) and other creatures using simulation, analysis, and visualization. https://devoworm.weebly.com/
[10] Deb, M., Singh, U., Deb, M., & Alicea, B. DevoLearn: Machine Learning Models and Education that Enable Computational Developmental Biology. https://github.com/DevoLearn/devolearn
[11] Tohsato, Y., Ho, K. H., Kyoda, K., & Onami, S. (2016). SSBD: a database of quantitative data of spatiotemporal dynamics of biological phenomena. Bioinformatics, 32(22), 3471-3479. https://ssbd.riken.jp/database/
[12] Cao, J., Wong, M. K., Zhao, Z., & Yan, H. (2019). 3DMMS: robust 3D membrane morphological segmentation of C. elegans embryo. BMC bioinformatics, 20(1), 1-13.
[13] Cell Tracking Challenge. http://celltrackingchallenge.net/
[14] EPIC dataset: https://epic.gs.washington.edu/
[15] WormWeb: http://wormweb.org/
[16] DevoZoo: https://devoworm.github.io/devozoo/index.html
[17] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[18] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
[19] TorchVision: https://pytorch.org/vision/stable/index.html
[20] Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., ... & Zhang, Z. (2019). Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315. https://www.dgl.ai/
[21] Amazon Shanghai AI Lab. https://www.amazonaws.cn/en/ailab/
[22] OpenHGNN: This is an open-source toolkit for Heterogeneous Graph Neural Network(OpenHGNN) based on DGL.
https://github.com/BUPT-GAMMA/OpenHGNN
[23] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637.
[24] Milan, A., Rezatofighi, S. H., Dick, A., Reid, I., & Schindler, K. (2017, February). Online multi-target tracking using recurrent neural networks. In Thirty-First AAAI conference on artificial intelligence.
[25] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. arXiv preprint arXiv:1801.01078.
[26] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
[27] Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.
[28] Song, X., Ma, R., Li, J., Zhang, M., & Wipf, D. P. (2021). Network In Graph Neural Network. arXiv preprint arXiv:2111.11638.
[29] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.
[30] Sigmoid activation function: https://en.wikipedia.org/wiki/Sigmoid_function
[31] Hyperbolic tangent (Tanh) activation function:
https://en.wikipedia.org/wiki/Hyperbolic_functions