Petascale Reconstruction of Neural Connectivity

Manuel Castro
Seung Lab
Princeton University
November 15, 2019
What is Connectomics?

- **Connectomics** is the study of how neurons are connected to each other in the brain.
- The brain’s neural connectivity, aka the connectome, can be thought of as a circuit. Connectomics aims to discover and understand these circuits.
What do we need to find neural connectivity?

- Given a volume of brain tissue, we must determine the shapes of the neurons within it, and the locations of their synapses (connections between neurons).
  - On the right, two neurons forming a synapse
Connectomics = Big Data

- In order to see the relevant structures, such as synaptic vesicles, brain tissue must be imaged at very high levels of resolution (e.g. 4x4x40nm).
- At this resolution 1 mm$^3$ of brain tissue contains $1.56 \times 10^{15}$ voxels (3D pixels). If a voxel is represented by a byte, that’s 1.4PB of data!
  - And that’s only 0.1% of the size of an entire mouse brain! (~1 cm$^3$)
Raw image collection
“Pinky” dataset (1μm^3)
Visualization by Nick Turner using Paraview
We have our image stack. Now what?

- (1) Segment the stack: Determine which voxels in the image stack belong to the same neurons. Equivalent to labelling each voxel with a neuron ID.
- (2) Detect where synapses are, and determine which neuron at each synapse is presynaptic and which is postsynaptic (synapses are directed).
- Two key insights: (1) we can train convolutional neural networks to perform (in part) these tasks; (2) these tasks are highly parallelizable.
Distributed Chunk Processing

- Divide the dataset into equally sized cuboid chunks
- Perform desired task on each chunk in parallel
  - Processing each chunk is independent
Distributed Chunk Processing

Image Data → Queue → Compute Nodes → Result
How to run inference: cloud or on-premise cluster?

1. For each dataset, only need to create segmentation and run synapse detection once
2. Need a lot of computation power to create this initial segmentation
   a. As many GPUs as we can get to run these ConvNets

Cloud computing, with its elasticity and scalability, is ideal for this kind of bursty workload.
Take advantage of preemptible/spot instances

1. Preemptible instances are ~3x cheaper than on-demand
2. Simple for us to make our processing robust to interruptions
   a. Auto-retry failed tasks
Example Architecture

Inspired by Wu et al. 2019

Thanks Kisuk Lee for use of his EM cube diagram.
GPU Comparison & Selection

- Interestingly, T4s are faster than P100s for us
  - Perhaps because of Tensor Cores on T4s, which are designed for deep learning inference ([https://developer.nvidia.com/tensor-cores](https://developer.nvidia.com/tensor-cores))
GPU Scaling

Scaling over time: a comparison of two datasets

- Pinky (1μm^3) using vCPU, 2017
- Minnie65 (0.65mm^3) using T4s, 2019

Affinity map inference on Google Cloud
Segmentation Procedure

- **Goal**: assign an ID to each voxel that represents which neuron it belongs to
- **Method**: ConvNet for boundary detection. Watershed for oversegmentation and mean affinity agglomeration for agglomeration.

Lee et al. 2019. *Convolutional nets for reconstructing neural circuits from brain images acquired by serial section electron microscopy*
Synapse Detection Procedure

- **Goal**: detect synapses and assign pre and post synaptic neurons.
- **Method**: ConvNet

Lee et al. 2019. *Convolutional nets for reconstructing neural circuits from brain images acquired by serial section electron microscopy*
3D Visualization
Proofreading & Analysis

● **Proofreading**
  ○ Problem: The automated segmentation is good but not perfect. Need humans to correct mistakes.
  ○ Goal: Dynamic segmentation system so humans can correct local errors in global segmentation
  ○ Method: Store segmentation as a graph.

● **Analysis**
  ○ Goal: Investigate neurons and discover cool things!

To make the above easier, we create meshes and skeletons for our segmentation.
Meshing & Skeletonization

- Create our own libraries for meshing & skeletonization
  - Existing libraries are applied to binary images
  - However, we want to mesh/skeletonize every object in each chunk
    - Inefficient to mask each object individually to create a binary image
  - zmesh (meshing library) & kimimaro (skeletonization library)

A few meshes of neurons in the FAFB (fly) dataset
One Wrinkle… Image Alignment

- Problem: in the collection process, tissue slices are often deformed and have defects
  - A tissue slice can have deformities like cracks or folds, causing discontinuities between subsequent slices
  - This makes it very difficult for a ConvNet or a human to reconstruct a volume
Solution: ConvNet

- ConvNet to align nearby slices
- Warp individual slices to minimize mean squared error between it and the few slices before it

Mitchell et al. 2019. *Siamese Encoding and Alignment by Multiscale Learning with Self-Supervision*
Solution: ConvNet

Before

After

Image stack (xz plane) pre and post alignment.
Alignment ConvNet is not as easily parallelized

- To align a stack of images, we need to start from a specific image and then warp each successive image
- Parallelizing across an image is easy, but parallelizing across images is hard
- Millions of small dependent tasks overwhelm some workflow management tools (e.g. Apache Airflow)
Summary of pipeline steps

1. **Deep Network Inference**: GPU intensive application of convolutional nets at scale. Easily parallelized, compute heavy.
   a. Image Alignment (not as easy to parallelize)
   b. Boundary Detection
   c. Synapse Detection

2. **Oversegmentation and agglomeration**: CPU and highly IO intensive application of classical algorithms. Requires a sequential hierarchy of increasingly larger instances.

3. **Interactive Proofreading System**: Use BigTable to store segmentation graph. Less data intensive, but requires low latency.
Live demo of Neuroglancer

Acknowledgements

This research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DoI/IBC) contract number D16PC0005, NIH/NIMH (U01MH114824, U01MH117072, RF1MH117815), NIH/NINDS (U19NS104648, R01NS104926), NIH/NEI (R01EY027036), and ARO (W911NF-12-1-0594). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government.

We are grateful for assistance from Google, Amazon, and Intel.

Thanks to Jeremy Maitin-Shepard for developing Neuroglancer and its associated data formats.