Third (7.23-7.29)

Summer Intern Work Report

Chen Xupeng

Supervisor:
Prof. Jeff Lichtman

July 30, 2018

Contents

1 Detailed progress and codes

I have create a category in my website to record more **Week4 Detailed Progress in Website**.

Also I have put all my codes in my github including four tasks: **NMJ, Synapse prediction, Synaptic partner, Synapse cluster**.

Project Codes1: All projects. **Project Codes2: NMJ automatic pipeline**

2 NMJ

2.1 Mask and seeding

Week4 NMJ Detail progress in website
NMJ codes
NMJ automatic pipeline codes

2.2 Thoughts about the whole project

This week I continue to seed on Mask3, and then I do a lot of exploration on how to do the NMJ project automatically.

I have understood how big and challenging this project is, it requires so many manually labeling work than we can't finish all the masking and seeding and segmentation and reconstruction work in two months. We know that it took KK and Marco several months to finish part of the bundle parts. But the remaining parts are more complex to seed, segment and it contains maybe 200 masks with approximately 50,000 sections. It is hard to estimate how long it will take to finish the whole project

However, as I am getting more familiar with this project, I am trying to build a more automatically pipeline for seeding, predicting membrane and segmentation. If it works, the project may move faster when we are here and after we leave:)

2.2.1 Seeding on Mask3

I felt that seeding on mask3 is much more complex than previous bundle seeding, the axon travels very fast and I should look up and down to look for one axon, it takes much more time to trace the branch than the main bundle.

2.3 Automatic pipeline

We would like to build up a more automatic pipeline before we leave and test the whole pipeline on several masks to see if they can be merged and reconstructed. We would like to

prepare all the codes and model for prediction and processing and write down the protocol.

The complete pipeline should contain:

- Generating Masks
- Seeding
- Predict Membrane
- Expand Seeds to get segmentation
- Merge different Masks

Previously we do seeding manually and then predict membrane, but the remaining masks have so many sections, I would like to do the seeding work more automatically too.

- **Predict Membrane**

  The automatically prediction parts must include membrane prediction, because it is "easier" to predict since the raw image already have the membrane.

- **Automatically seeding**

  The traditional way is to manually put seeds on each axon, but we have approximately 50,000 sections if all masks are generated, it is so time-consuming to manually put seeds. I will generate seeds by **distance transformation from membrane.**

  Then the seeds must be indexed to track each seed is from which axon, so we will manually put seeds per 100 sections, then do **Hungarian matching**.

- **Segmentation**

  Expand the seed to generate segments

- **Merge masks**

  We are thinking about linear interpolation to merge anchor sections for loop problems. We will discuss it more with Daniel and Yaron after the segmentation

2.4 Algorithm

The related codes are here: **NMJ automatic pipeline codes**

2.4.1 Predict Membrane

This week I create the membrane prediction pipeline using 3D U-net model, it is more powerful than previous membrane prediction model.

The 3D U-net model to use contours extracted from dense segmentation sections. Use 50 sections for training, then predict more, proofread predicted sections to generate more training samples. **The iterative training and predicting method will make the model more precise.**

The model's weight is adaptive to the pixels ratio, I can do fine tune on the model iteratively. So the model will be more precise and requires fewer proofreading. Last week I do many augmentation works, it is also useful to generate more training images since I only have 50 sections for training now.

How to fine tune:

If we want better result, we can manually label several sections on each mask and retrain the model on each mask.

For membrane prediction, since we do not consider affinity, we can also consider 2D U-net, it contains much less parameters and easier to train.

2.4.2 Automatically seeding

- **Distance transformation** to generate seeds from membrane
- **Hungarian matching** to label each seeds for different axons. Manually label one section' s seed and do Hungarian matching for the next 100 sections.

2.4.3 Watershed

Use watershed to expand seeds and generate segment

Other possible algorithm:

Use EM data to predict seeds, train seeding prediction network, using affinity because the axon travels fast. Sebastian's group has some work. But I think it is imprecise compared to membrane prediction&distance transformation algorithm

2.5 Pipeline on membrane prediction

2.5.1 Prepare ground truth training set

I have started on membrane prediction pipeline after discussion with zudi, yaron and others. I would like to use previously label siyan and I have done in first two weeks to save time. We have done dense segmentation on 51 sections, I wrote a python script: create gound truth codes to extract the needed EM image and contours of the membrane in the following steps:

- export segmentation and EM ROI from VAST
- read in python, converting id array to RGB array for visualization. [fig1]
- find bounding box of each segmentation and EM image.
- remove Schwann cell to concentrate on axons. [fig2]
- convert the segment array to binary mask. [fig3]
- Make sure each mask and EM data are in same bounding box. [fig4]
- Generate contours as training set label. [fig5]
- Padding for same image size, Do reflection padding on each image to generate images with same size. [fig6]
- Store in HDF5, EM data as training set's image and contour as label. Save as uint8 (51, 530, 835)

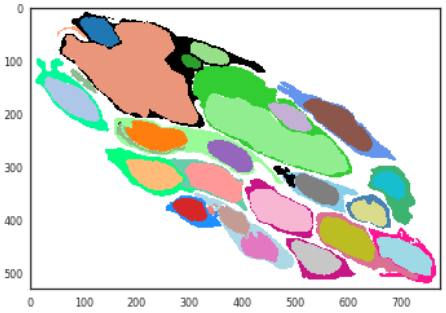

Figure 1: RGB visualization

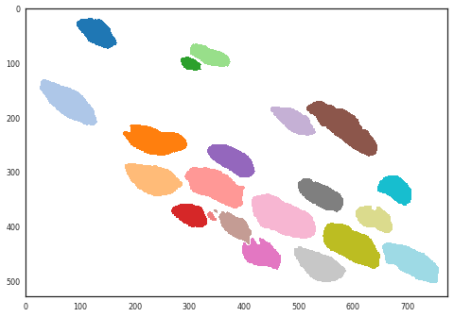

Figure 2: remove Schwann cells

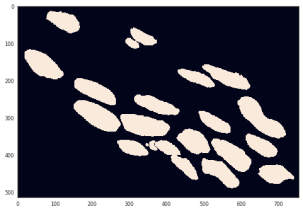

Figure 3: binary mask

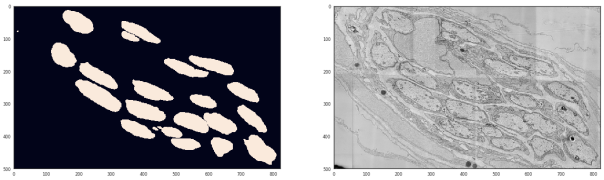

Figure 4: same bounding box

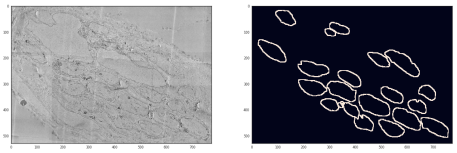

Figure 5: Generate contours

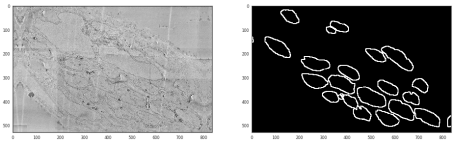

Figure 6: padding

2.5.2 Train membrane prediction model

Input image and label are the 51 bundle sections. Run on two machines: one with one gpu and another with 4 gpus.

Use tensorboard to check loss. Monitor loss and test on new EM image. If is good, train it longer with more GPUs.

Loss Function:

$$L = 1 - \frac{2 \times \sum_{i=1}^{N} |P_i \cap GT_i|}{\sum_{i=1}^{N} (P_i + GT_i)} + \sum_{i=1}^{N} FocalLoss(P_i, GT_i)$$

$$FocalLoss(pt) = -(1 - p_t)^{\gamma} log(p_t)$$

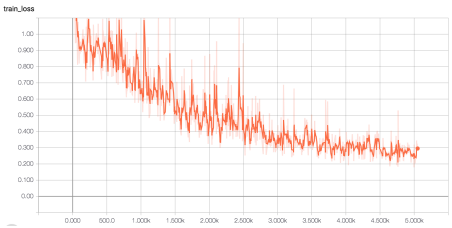

Figure 7: DICE+Focal loss

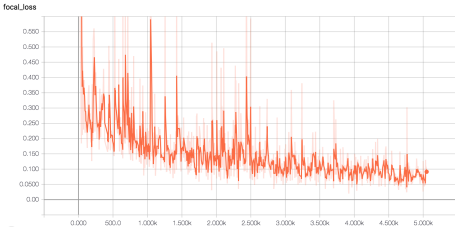

Figure 8: Focal loss

It seems that the combined loss and both focal and dice loss decrease well.

real time monitoring predicted result Use TensorboardX to monitor predicted results, This will allow me to see the improvement of model's performance more clearly. (fig9, 10, 11)

Then I will predict new EM image which is preprocessed by the previous steps. Then do proofreading on the predicted membrane. Then do distance transformation to generate seeds.

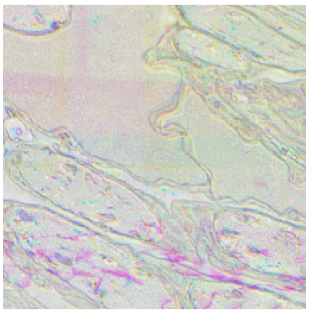

Figure 9: EM in 3680th batches

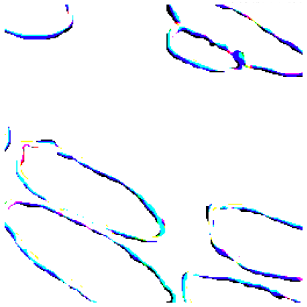

Figure 10: Ground truth in 3680th batches

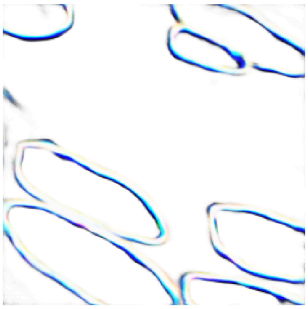

Figure 11: Predicted in 3680th batches

3 Synapse Prediction

Week4 Synapse Prediction Detail progress in website
Synapse prediction codes

This week I mainly focus on synaptic partner prediction and NMJ pipeline, and wait for our current models' result on CREMI synapse prediction challenge.

3.1 Loss function improvement

Last week we add DICE loss and this week we change BCE loss to Focal loss. It is adaptive to better consider weights.

P: predict result, GT: ground truth, N: batch size

$$\textbf{FocalLoss}(p_t) = -(1-p_t)^{\gamma} log(p_t)$$

$$L = 1 - \frac{2 \times \sum_{i=1}^{N} |P_i \cap GT_i|}{\sum_{i=1}^{N} (P_i + GT_i)} + \sum_{i=1}^{N} FocalLoss(P_i, GT_i)$$

3.2 Augmentation improvement

This week I add and improve two augmentation methods: deform and elastic augmetation. augmentation codes

3.2.1 deformation augmentation

Remove random seeds, do not add deformation on masks. Write the function to avoid deformation on adjacent sections. We wish this can force the network pay attention to 3D characteristics of the image. (fig12)

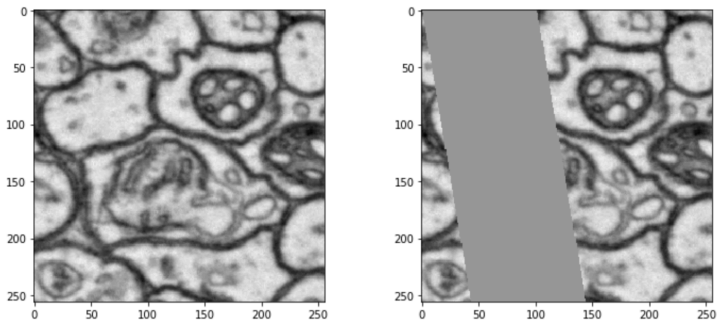

Figure 12: deformation augmentation

3.2.2 elastic augmentation

elastic augmentation is important and should be treated very carefully. I use the idea in Best Practices for Convolutional Neural Networks applied to Visual Document Analysis.

The main methods are Gaussian filter and interpolation. There is a version of elastic augmentation using affine transformation, I test it but the effect is hard to control.

So I use Gaussian filter and interpolation for gentle elastic transformation. Then normalize to 0-1, binarize label and make sure the transformation is same in images and masks. (fig13)

3.3 task2 reverse predicting

I have use alignment and padding and shift scripts to process the raw data and it should be reversed after prediction. I test the codes to make sure it works.

reverse align codes

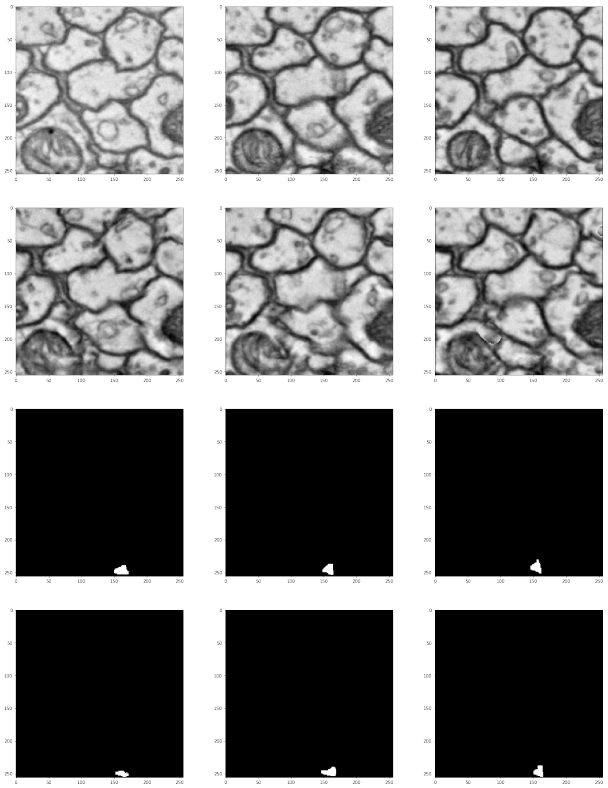

Figure 13: deformation augmentation

4 Synaptic Partner

Week4 Synaptic Partner Detail progress in website
Synapse partner codes

This week we mainly focus on task 3, synaptic partner prediction, this is a new and more challenging task for us. Previously we never try to predict synaptic partner, so we have to understand this task and process the raw data to get the train label.

4.1 Align and process location to create training labels

I study and understand the synaptic partner's identification criterion and process the location value in the same alignment steps with raw and clefts image.

- extract location and pre-post id, put these points in a volume
- align the volume, padding and deal with bad slices using same strategy

location is only point id, we can shift the point by first draw the point in zero array and then extract location. Each point use pre and post id to trace back. The premise is location and clefts matches, I have checked it.

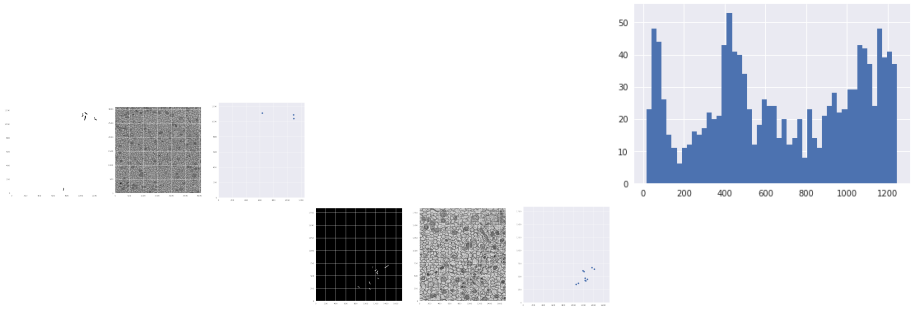

Figure 14: visualize shift results

Codes: alignment matlab script This script can align raw, clefts and location and reverse them.

Deal with pre and post processing of locations array.

I also find that in A, B, C three volumes, the partner numbers differ a lot:432, 1324, 2276. But clefts numbers are similar:123, 131, 165.

The align work needs many patience, I did it over ten times to ensure it is 100 percent right. The axis, scatter and imshow tradition and many steps process made it easy to make mistakes. At last I found that it is best to dilate the points in location array and visualize it using imshow uniformly to check the alignment.

4.2 Synaptic partner identification model

The model can be derived from synapse identification model. We also use 3D U-net and add some changes. We study Funke's previous work, it has some strengths and drawbacks.

For example, they use 14 vectors to represent all possible partner vectors. And the model predict a 14 channel one-hot vector. But this methods deliberately overfits on CREMI datasets. Since we would like to develop a model to predict synaptic partners on JWR datasets, we should use a more generalized methods. But the searching space will increase a lot.

- dilate location point

  for each points(so it can specify location)

- Generate 4 channel output (binary, Z, Y, X)

  binary for **localization**, and Z, Y, X for vector **orientation**. We will use cosine loss to evaluate orientation. So this can be considered as a multi task training

- use branched model to predict two parts

  Output mask and vector separately. A very natural thought is we can design the model further to multiply the binary channel to the orientation part as **Attention mechanism**, the two branched models can share weights. I use a similar architecture in another project and it works well.

It is hard to both predict orientation and length. So maybe we can do post-processing work: we only predict orientation, and calculate the distance from pre synapse to clefts, and add the distance to produce post synapse location.

4.3 Hacking toufiq's codes

I also try to study and understand Toufiq's work. I found Toufiq and Lee has done a lot on task 3 and related synapse work. They have some valuable work and codes to be recovered. So I try to find something useful to use. I still have a lot of work to do since Toufiq wrote a huge amount of codes.

synapse-polarity codes

Prediction of pre and post and segmentation(pre and post is self-labeled and segmentation is predicted) and clefts(ground truth in original dataset) (fig17)

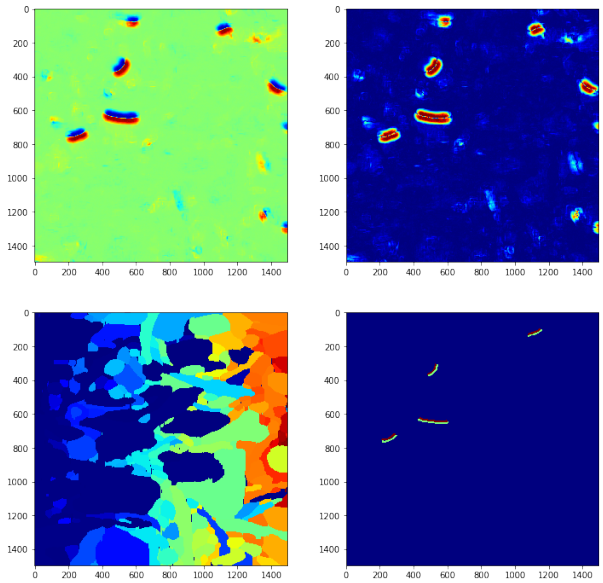

Figure 15: deformation augmentation

5 Synapse Cluster

Week4 Synapse Cluste Detail progress in website
Synapse cluster codes

Help siyan with 3D skeleton. Direct 3D skeleton isn't very good since the distance of sections (30/40 nm) is a little long. Direct interpolation or dilation also works badly.

So I try a ICP and KNN algorithm to maximize matching of two 2D contours and find the nearest neighbor of each point in two neighbor contour. This idea originate from calculating the volume size of a 3D object. After finding each points neighbor, we can do linear interpolation to create another 9 sections for better 3D skeleton.

Codes: KNN interpolation

6 References

[1] Turaga, Srinivas C., et al. "Convolutional networks can learn to generate affinity graphs for image segmentation." Neural computation 22.2 (2010): 511-538.

[2] Felzenszwalb, Pedro F., and Daniel P. Huttenlocher. "Efficient graph-based image segmentation." International journal of computer vision 59.2 (2004): 167-181.

[3] Briggman, Kevin, et al. "Maximin affinity learning of image segmentation." Advances in Neural Information Processing Systems. 2009.

[4] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." arXiv preprint arXiv:1708.02002 (2017).

[5] Simard, Patrice Y., Dave Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." null. IEEE, 2003.