Second (7.16-7.22)

Summer Intern Work Report

Chen Xupeng

Supervisor:
Prof. Jeff Lichtman

July 22, 2018

Contents

1 Detailed progress and codes

I have create a category in my website to record more **Week3 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 Codes**

2 NMJ

2.1 Mask and seeding

NMJ Detail progress in website
NMJ codes

This week we have worked out a plan on alignment and seeding.

We use a mind map to record tree's nodes to visualize our progress. The mask was sent to Adi for aligning. The alignment results seem very good.

I have seeded three masks Adi sent back, **for about 900 sections mainly in bundle area.**

I also wrote python script plot seeding for further analysis. Since I am proficient in using python for visualization, statistical analysis and machine learning, I wrote some python scripts to read seeding result, visualize them and plot them in 3D and animation. It will be better to have more statistical analysis when I collect more seeding data.

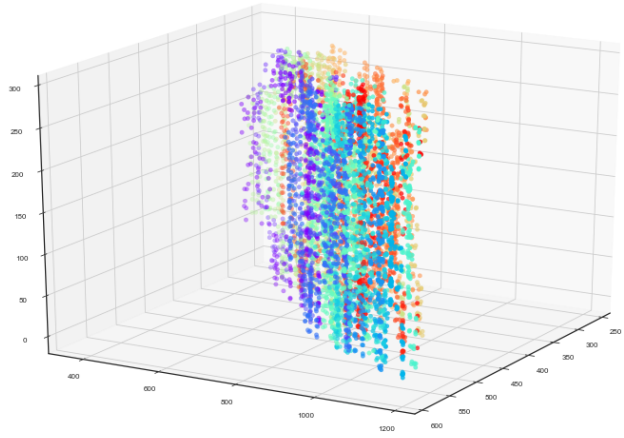


Figure 1: seededng visualization

2.2 Discussion on masks

When we put masks on ROI, we found many branches even in bundle area, two branches from one stem may encounter and form a closed loop. We are worried if it will be a problem when we merge all masks together. After discussing with Adi and Daniel, we understand that the spatial structure's change isn't a big problem.

3 Synapse Prediction

Week3 Synapse Prediction Detail progress in website
Synapse prediction codes

In week 3, apart from NMJ project, I concentrate mainly on synapse prediction project and make many progress. Since zudi has come back, it is more efficient to discuss and collaborate, we have done a lot this week.

3.1 short term plan

Our short term plan is adding all the new tricks (dilation CNN block from Deepglobe's $Dlink-net$, $DICE-BCE$ combined loss, all the data augmentation methods) to the model. We train the whole model for three days on all A, B, C samples. And fine tune it later for one day. Then we submit the result to see if it is better.

In my consideration, the previous model **has a not very little gap with the state-of-art one**. About one fold error score. So it is a lot for us to do. **I have used dilation CNN, combined Loss, and some very useful data augmentation methods** which take me many days to accomplish(All admit that proper augmentation is one of the most import procedure to achieve better results.) So this week I have read many other people's good paper, project summary and codes in similar tasks(**the dilation CNN,** $DICE-BCE$ **loss and data augmentation all originate from this.**) I believe there are more to implement. The $state-of-art$ model and pipeline looks really good, so I think I should use more strategy to improve our results.

3.2 Model improvement and training

3.2.1 settings and training

Since I have done several deep learning and machine learning projects(Emaize, CardiacAI, 3D CT images, Deepshape), I am quite familiar with linux, python deep learning environment and all extra settings(jupyter, TensorboardX etc). I can be quite efficient in running the previous model designed by zudi.

- training settings

- create envs
- training parameter setting
- Tensorboard monitor setting

3.2.2 network visualization

I visualize our current model, it is a very big one.
Scripts: pytorch synapse.ipynb

3.2.3 model structure and loss function improvement

I read D-LinkNet: LinkNet with Pretrained Encoder and Dilated Convolution for High Resolution Satellite Imagery Road Extraction and other winners in some challenges and implement more model structure and loss function design.

dilated CNN block It can further enlarge the visual area of the model, which is helpful to learn more information efficiently. The dilation is adaptive according the the layer depth. The maximum is 8, we may change it smaller later.

loss function I read Deepglobe challenge's solution and change the loss function to **DICE + BCE** like this: P: predict result, GT: ground truth, N: batch size

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

It added a DICE part to previous BCE loss function, which is better since it is common to use DICE as loss function for U-net, the combined loss function is also used in some challenges winner.

I use TensorboardX to monitor the training procedure. I monitor the combined loss, DICE loss and BCE loss simultaneously. To my relief, the loss function decrease not only because BCE decrease, the DICE loss also decrease. Since 1- DICE reflect the overlap ratio of predicted and ground truth area, it means our model learns well.

Train loss

We should do further improvement on loss function, BCE is divided by batch size, so we may do the same with DICE.

We think BCE punishes FN since we set a weight to punish it. And we think DICE is punishing FP since the remained are mainly FP, and we think it is good: our strategy is remove FN at first, then we can prune FP using some methods. So DICE loss may become one of our methods.

We can also balance the FP, FN by adjusting DICE and BCE's weight. It is essential for us to decrease FP since it is the main gap between the state of art model. We retain so many FP in predicting.

3.3 Data augmentation

I spent many time on this part because I believe it is the most important part besides the model, also this part may determine which pipeline is better between many good models.

Last week we have discussed a lot about augmentation's details. This week I rewrite several of them, and learned many different augmentation methods from many sources(mature pipeline, API, challenge winner and papers) I compare their results by rewrite their codes and visualize the results. Some of the codes I read are hard to understand(like gunpowder) since it contain a whole pipeline's complex manipulation.

I have applied several of them and implement them in training and testing pipeline.

The codes for final use is: augmentation.py

And the implementation and visualization codes: cremi augmentation implementation augment cremi

3.3.1 training augmentation

I do transformation on image and mask at the same time. For simple augmentation it is simple, for elastic transformation, I use random seed to reproduce, and binarize mask data to 0, 1 again. For intensity augmentation I don't do any transformation on mask. For defect transformation, there are many to notice, including random seed to reproduce and binarize.

I do many visualization to benchmark every augmentation

simple augmentation Do X, Y, Z mirror and transpose, so in total there are 16 methods. Produce a list to generate four 0, 1 random int to decide do or not do these four methods. Do same transformation on image and mask.

intensity augmentation Use **mix** mode for random choose 2D or 3D intensity augmentation. I do not transform mask because there is only pixel shift.

elastic augmentation I compare gunpowder and this kaggle elastic one, after comparison, it seems gunpowder version elastic has more functions, if we do not use rotate, it is fine.

The output range of create transformation is $0 - width$, we can normalize the results. And I use random seed to reproduce on images and masks

defect augmentation I use gunpowder's defect, the block region used to be zero, but I change it to zero for better batch normalization results. I use random seed to reproduce images and masks. The block size and missing section's ratio can change.

3.3.2 test augmentation

For each sample in test, I will produce 16 transformed images and reverse the predicted masks later.

In test, it is common to only use 16 kinds **simple augmentation**

For X Y Z axis mirro and xy transpose, in all $2^4 = 16$ images produced.

3.3.3 further useful augmentation

I have found many other augmentations in many sources. Since our main aim now is to get a better result in a short time. I will see if the augmentation strategy is enough. If not, I will implement them later. If the augmentation is good enough, we can do more on task 3 later. Since the model is really hard to train and see the results(may take a week for the feedback), time is really precious, and we can't test one method at one time.

I have tried some here further augmentation, but I will decide what to do first next week.

- 2018 annual datascience bowl top1: segment cells. There are many data augmentation methods: It seems clear and they used an augmentation API imgaug. Apart from gunpowder's method, there are a lot to use!
- random zoom, rotate, flip
- contrast and brightness
- heavy geometric transform: Elastic Transform, Perspective Transform, Piecewise Affine transforms, Pincushion Distortion
- contrast limited adaptive histogram equalization (CLAHE) Sharpen Emboss
- Gaussian noise Blur Median Blur Motion Blur
- HSV rearrange channel repeat of cell nuclear

3.4 future work

3.4.1 compare with other work and thoughts

- For randomly chosen sample, we use pixel ratio to determine if one sample is used to train, Funke use probability, we can think which is better.

- Funke use regression whereas we use binary classification, it is simple to change to regression, but we should compare our loss function and strategy at first. For example, Funke's STED loss may not be good.
- Funke use auxiliary loss for better prune, but we have visualize the result and find the proposed synapse matchh the membrane well, so it may not be necessary.

3.5 future plan

As mentioned above, I believe there are more to implement. The state-of-art model and pipeline looks really good, so I think I should use more strategy to improve our results.

- read and consider V-net's structure. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation
- if submitted result has a big difference with Funke's, I may try to reproduce their work.
- read task 3 complete codes apart from previous postdoc's codes.
- how to fine tune, apart from learning rate adjustment, we should consider some prune work. For example, the paper **Stacked U-Nets with Multi-Output for Road Extraction**
- examine results carefully to understand the difference between FP and TP, gain biological intuition from it for better model design. For example the density of membrane and vesicle

4 Synaptic Partner

Week3 Synaptic Partner Detail progress in website
Synapse partner codes

I have read and summarized some paper and codes last week on synaptic partner project. This week zudi and I discuss about previous work's strategy. We have decided that I read and study the synaptic partner codes written by a previous postdoc.

Now the codes is incomplete in github, later we may get the complete codes. The codes isn't in a very good structure, and task 3 is harder and more complex than synapse prediction, so it will take me some time to fully understand the codes and make more improvements on it.

5 Synapse Cluster

Week3 Synapse Cluste Detail progress in website
Synapse cluster codes

I have further considered **clustering** project. Since we will have a huge amount of data to cluster, it is a natural thought to use deep learning based clustering method, which I have mentioned last time. After discussion with donglai and zudi, they also agree we can use a (variational) auto-encoder to cluster and analyze the synapse. If have time, I may try 3D VAE to cluster the synapse, but it need some preprocessing work. We may at first align and rotate the 2D image for better results.

I have read some papers including **Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling**, **Clustering with Deep Learning:Taxonomy and New Methods**. Which I thought will be useful

6 References

[1] Wu, Jiajun, et al. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." Advances in Neural Information Processing Systems. 2016.

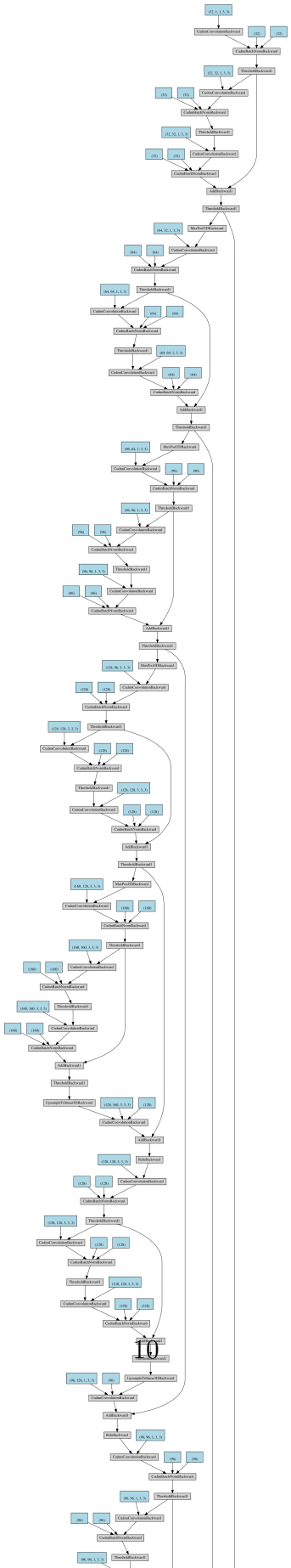
[2] Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." 3D Vision (3DV), 2016 Fourth International Conference on. IEEE, 2016.

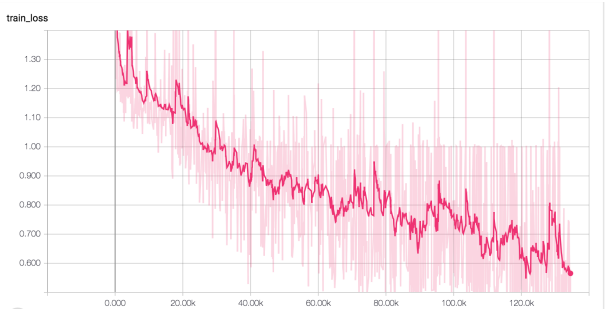
[3] Zhou, Lichen, Chuang Zhang, and Ming Wu. "D-LinkNet: LinkNet with Pretrained Encoder and Dilated Convolution for High Resolution Satellite Imagery Road Extraction." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018.

[4] Sun, Tao, et al. "Stacked U-Nets with Multi-Output for Road Extraction." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018.

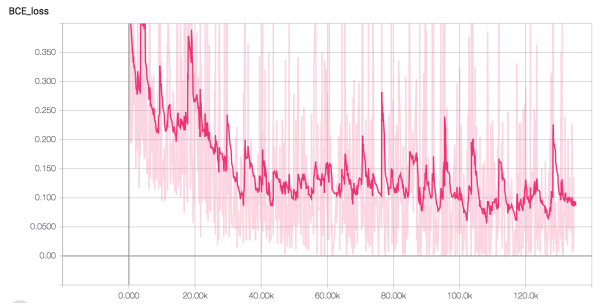
[5] Ding, Jiarui, Anne Condon, and Sohrab P. Shah. "Interpretable dimensionality reduction of single cell transcriptome data with deep generative models." Nature communications 9.1 (2018): 2002.

[6] Aljalbout, Elie, et al. "Clustering with Deep Learning: Taxonomy and New Methods." arXiv preprint arXiv:1801.07648 (2018).

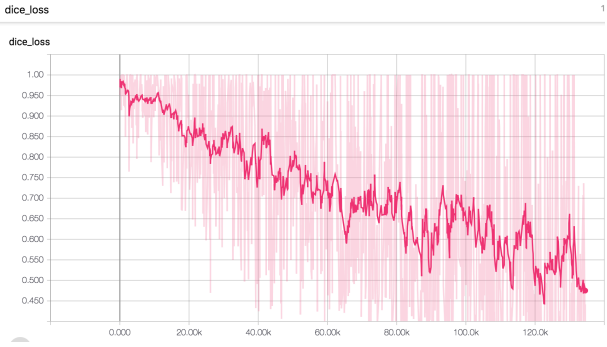

(1, 1, 3, 3, 3)
CubeConvolutionBackward2
(32)
(32)
CubeBackNormBackward
(1, 32, 1, 3, 3)
ThresholdBackward0
(32)
(32)
CubeConvolutionBackward2
CubeBackNormBackward
(32, 32, 1, 3, 3)
ThresholdBackward0
(32)
(32)
CubeConvolutionBackward2
CubeBackNormBackward
AddBackward0
ThresholdBackward0
(64, 32, 1, 3, 3)
MaxPool3DBackward
(64)
(64)
CubeConvolutionBackward2
CubeBackNormBackward
(64, 64, 1, 3, 3)
ThresholdBackward0
CubeConvolutionBackward2
(64)
(64)
CubeBackNormBackward
ThresholdBackward0
(64, 64, 1, 3, 3)
CubeConvolutionBackward2
(64)
(64)
CubeBackNormBackward
AddBackward0
ThresholdBackward0
(96, 64, 1, 3, 3)
MaxPool3DBackward
CubeConvolutionBackward2
(96)
(96)
CubeBackNormBackward
(96, 96, 1, 3, 3)
ThresholdBackward1
CubeConvolutionBackward2
(96)
(96)
CubeBackNormBackward
(96, 96, 1, 3, 3)
ThresholdBackward1
CubeConvolutionBackward2
(96)
(96)
CubeBackNormBackward
AddBackward0
ThresholdBackward0
(128, 96, 1, 3, 3)
MaxPool3DBackward
(128)
(128)
CubeConvolutionBackward2
CubeBackNormBackward
(128, 128, 3, 3, 3)
ThresholdBackward0
CubeConvolutionBackward2
(128)
(128)
CubeBackNormBackward
ThresholdBackward0
(128, 128, 3, 3, 3)
CubeConvolutionBackward2
(128)
(128)
CubeBackNormBackward
AddBackward0
ThresholdBackward1
(160, 128, 3, 3, 3)
MaxPool3DBackward
CubeConvolutionBackward2
(160)
(160)
CubeBackNormBackward
(160, 160, 3, 3, 3)
ThresholdBackward0
CubeConvolutionBackward2
(160)
(160)
CubeBackNormBackward
(160, 160, 3, 3, 3)
ThresholdBackward1
CubeConvolutionBackward2
(160)
(160)
CubeBackNormBackward
AddBackward0
ThresholdBackward1
UpsampleTrilinear3DBackward
(128, 160, 3, 3, 3)
(128)
CubeConvolutionBackward2
CubeBackNormBackward
AddBackward0
ReluBackward
(128, 128, 3, 3, 3)
CubeConvolutionBackward2
(128)
(128)
CubeBackNormBackward
(128, 128, 3, 3, 3)
ThresholdBackward0
CubeConvolutionBackward2
(128)
(128)
CubeBackNormBackward
ThresholdBackward0
(128, 128, 3, 3, 3)
CubeConvolutionBackward2
(128)
(128)
CubeBackNormBackward
10
(96, 128, 1, 1, 1)
(96)
UpsampleTrilinear3DBackward
CubeConvolutionBackward2
AddBackward0
ReluBackward
(96, 96, 1, 1, 1)
CubeConvolutionBackward2
(96)
(96)
CubeBackNormBackward
(96, 96, 1, 3, 3)
ThresholdBackward0
CubeConvolutionBackward2
(96)
(96)
CubeBackNormBackward
(96, 96, 1, 3, 3)
ThresholdBackward0



(a) DICE BCE loss



(b) BCE loss



(c) DICE loss

Figure 3: Loss overview

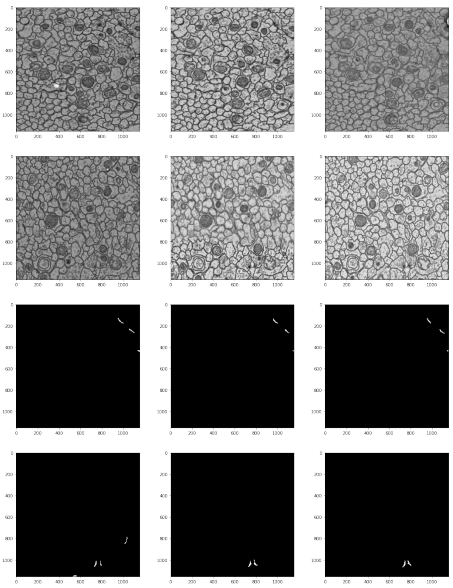


Figure 4: simple augmentation

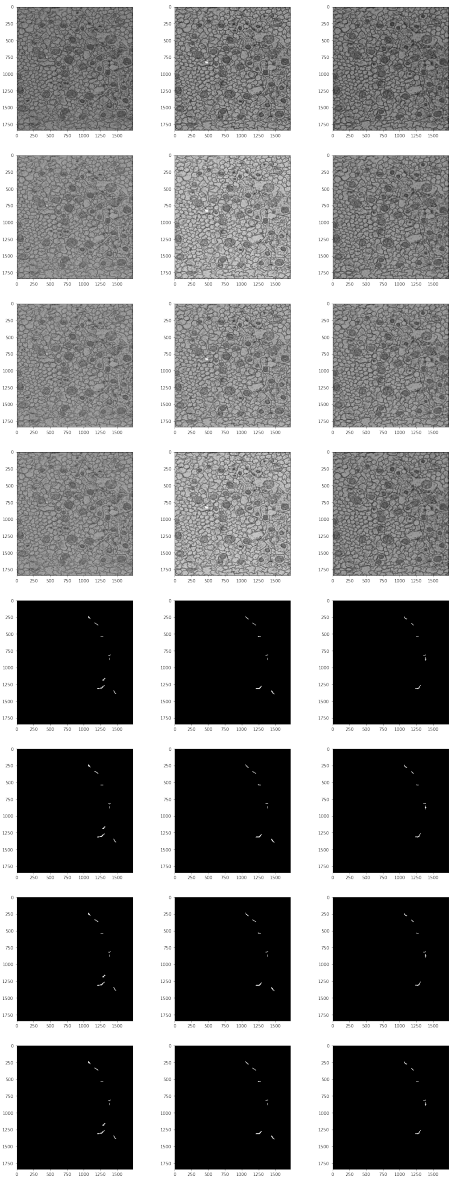


Figure 5: intensity augmentation

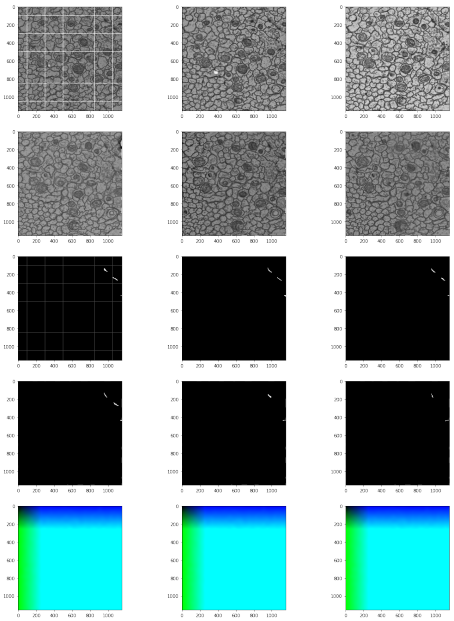


Figure 6: elastic augmentation

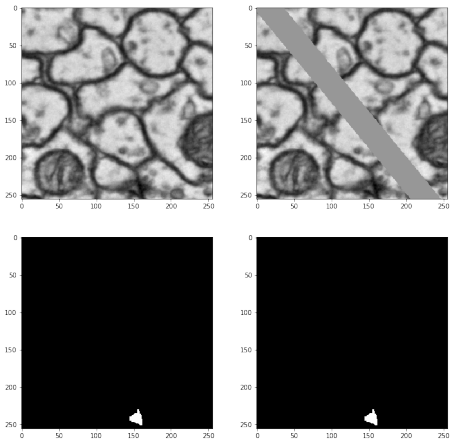


Figure 7: defect augmentation