Modified u-net

1. Network Architecture

Pros & Cons to use FCN/U-net:

Pros:

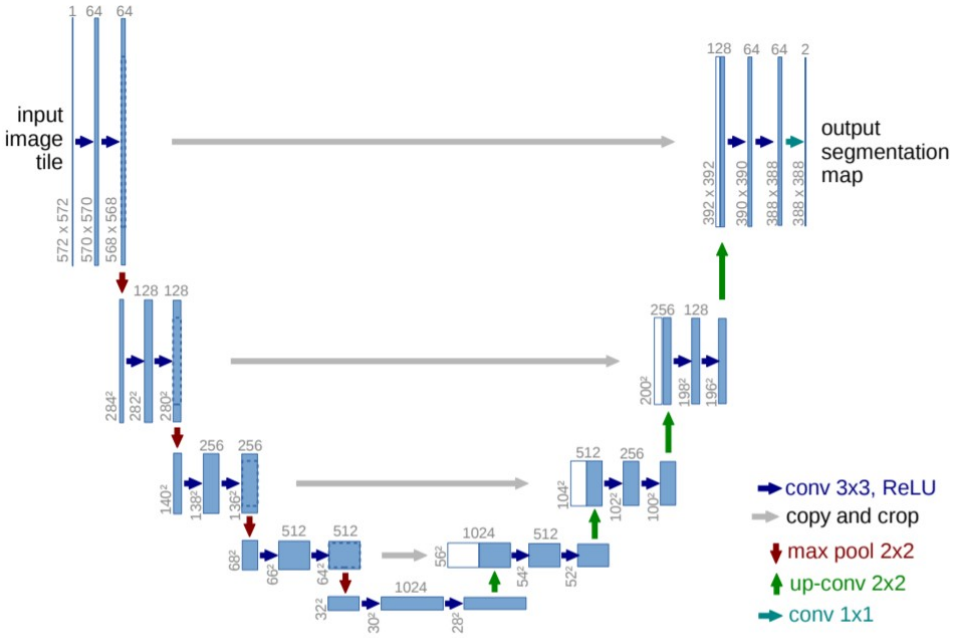
1. Able to learn high level/abstract features (compare to traditional CV method & machine learning) In this task, model expected to learn to distinguish different features like cell nucleus, membranes, noise etc. so that the model can enhance some of them while dispose others.
2. Reserve positional information (compare to CNN with FC layers) so that the model can output predicted mask.
3. (U-net) Using skip connections can help make network deeper, and further reserve the positional information…...

Cons:

1. Requires large amount of data to train, but got only 30 pairs of training data.
2. Difficult to find optimal hyper parameters
3. Very computationally/ space expensive makes training extremely slow and inefficient…...

Network structure:

The main idea of the method is to modify the original u-net structure proposed in [1].



There are three primary modification:

1. Replace all 2x2 max pooling layers with 2x2 convolution layers with 2x2 stride.

According to Geoffrey Hinton[6]: “The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster.”

Pooling layer is widely used in CNN. It works well on reducing the computational as well as memory cost by significantly reduce the size of feature map (e.g. size /= 2 after apply 2x2 pooling). Another important effect is: pooling layers can increase receptive field and enhance in-variance. However, pooling layers can lose detailed positional information, which is pretty important for image segmentation tasks. Last but not least, in my understanding, pooling is intuitively not what the brain does to work on images.

In another point of view, pooling layers are nothing but a set of untrainable, fixed-rule convolution layers. For example, a 2x2 max pooling layer reduce the feature map size to half(down sampling),

and remarkably increase receptive field. That also can be done by a 2x2 convolution layer with stride=(2,2). The difference is that the former is untrainable, the latter is trainable.

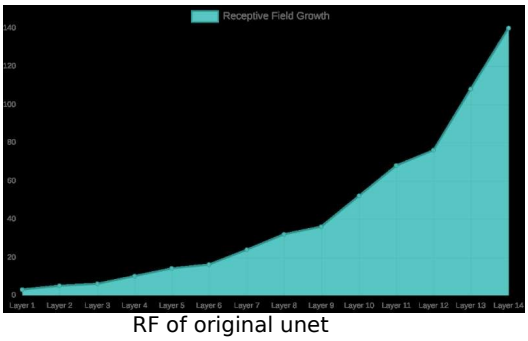
2. Introduce dilated convolution[4] to gain larger receptive field while reserve positional information.
By replacing the second 3x3 convolution layer, in each down sampling step, to a convolution layer with dilation rate=3, RF is growing faster.
To avoid 'checker-board artifact' (aka 'the gridding effect') and to keep sensitive on short-ranged information, the model still keeps one convolution layer without dilation so that every pixel would be covered[5].

Checker-board artifact

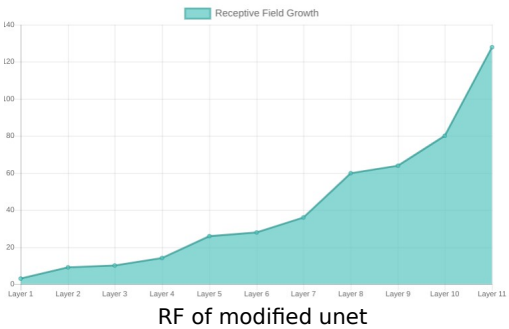
The receptive fields of original u-net and new model are calculated at [7], shown below:

Layer #	Kernel Size	Stride	Dilation	Padding	Input Size	Output Size	Receptive Field
1	3	1	1	0	572	570	3
2	3	1	1	0	570	568	5
3	2	2	1	0	568	284	6
4	3	1	1	0	284	282	10
5	3	1	1	0	282	280	14
6	2	2	1	0	280	140	16
7	3	1	1	0	140	138	24
8	3	1	1	0	138	136	32
9	2	2	1	0	136	68	36
10	3	1	1	0	68	66	52
11	3	1	1	0	66	64	68
12	2	2	1	0	64	32	76
13	3	1	1	0	32	30	108
14	3	1	1	0	30	28	140

Layer #	Kernel Size	Stride	Dilation	Padding	Input Size	Output Size	Receptive Field
1	3	1	1	0	688	686	3
2	3	1	3	0	686	680	9
3	2	2	1	0	680	340	10
4	3	1	1	0	340	338	14
5	3	1	3	0	338	332	26
6	2	2	1	0	332	166	28
7	3	1	1	0	166	164	36
8	3	1	3	0	164	158	60
9	2	2	1	0	158	79	64
10	3	1	1	0	79	77	80
11	3	1	3	0	77	71	128



RF of original unet



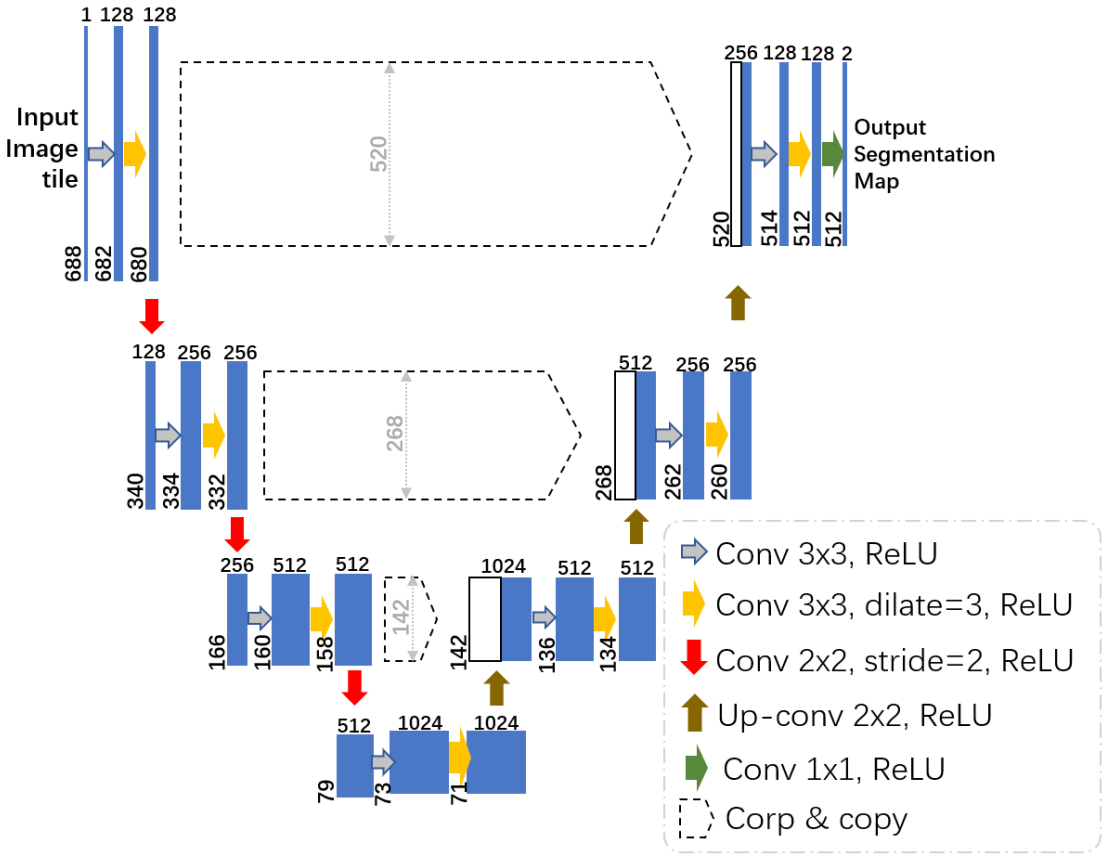
RF of modified unet

original unet uses 14 layer of down sampling to reach the RF of 140, on 30x30 pixels, while modified unet uses only 11 layers of down sampling to reach the RF of 128 but on 71x71 pixels, that indicates modified unet has more features pixel to hold more precise features than the original one.

3. Reduce the depth of the network.

Unet has 4 steps of down sampling, modified unet has 3. That is mainly to reduce the computational and memorial cost during training.

The final model is shown below:



2. Experiment

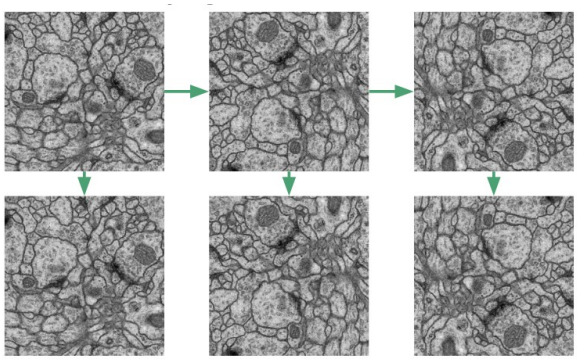
the augmented data set is used to train the network with Adam optimizer implementation by tensor flow core. All weights in network are initialized by Xavier normal initializer and bias are initialized as all zeros.

2.1 Overlap-tile strategy

As all convolution layers in the model use padding=’valid’ (i.e. no padding). To get the result of the same size with the input & label, input image is up scaled and missing pixels are extrapolated by mirroring.

2.2 Image augmentation

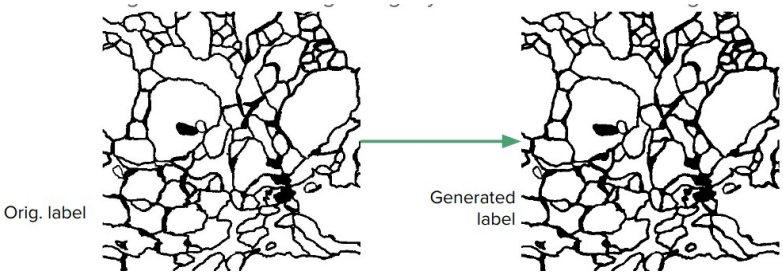
Applied data augmentation by rotation, flip, and elastic deformation on 83%(25 images) of the original development data set. The elastic deformation follows the method in [3]. augmentation demo shown below ,where downward arrow indicates elastic deformation.



However, apply same elastic deformation directly to label images would sometimes results in jagged edges and broken morphological properties, such as non-closed membrane and solid small cells. To smooth the edge and recover these properties, I do the following procedures:

elastic_deform → Gaussian_filter → threshold(thresh>127) → label_result

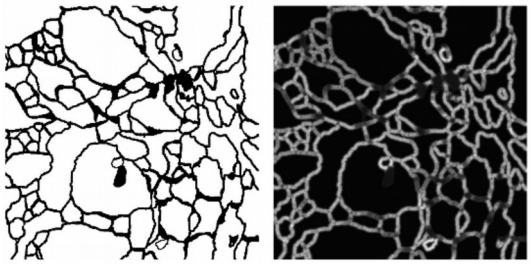
Gaussian filter to smooth the edges, and thresholding with thresh>127 recover the properties.



That lead the membrane in generated label a little bit thicker than the original label, but it's not a big problem.

2.3 Weight maps

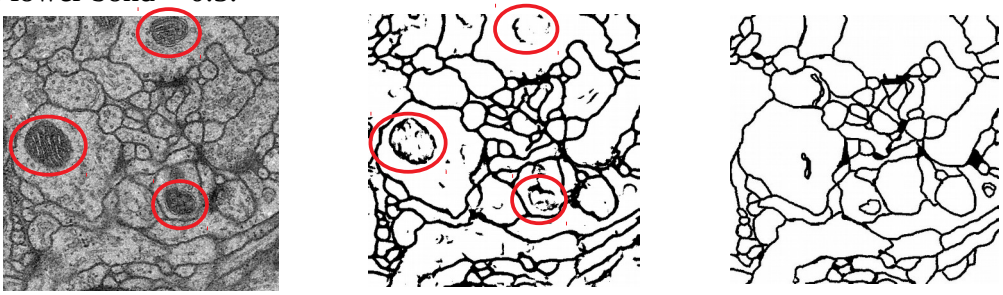
To force the networks to learn the thin membrane in the label masks, I used generated weight map to combine with cross entropy loss function follow the method mentioned in [1].



For each label in training set, a weight map has to be generated. There was total 600 training data generated in the experiment(off-line data augmentation).

During the training, what the model was expected to learn is both enhance the membrane and remove nucleus and noise. However, weight maps assign higher weight only on thin edges, that makes the model only focus on not losing membrane, but the process to learn to remove nucleus very slow.

To speed up the process of 'learning to remove', all weights in a weight map whose value is smaller than a constant called lower bond, is scaled up to equal to lower bond. In experiment, lower bond was set as lower bond = 0.5.



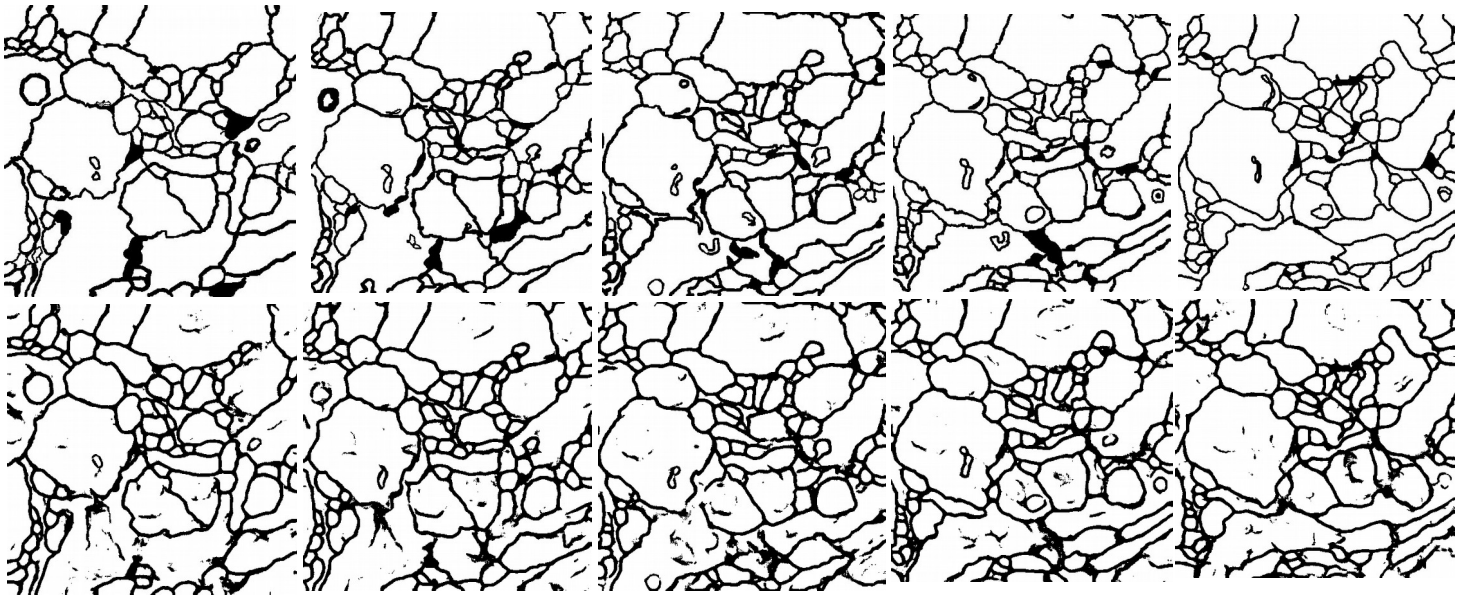
As shown above, a result (middle) produced after 1000 iteration of training without setting weight lower bond. The model fail to remove the nucleus in the red circles.

2.4 why not do cross validation

1. Takes too long to training one model (1500 iteration) and to make weight map(48h made 600 weight maps)
2. Overfitting won't happened in 1500 iteration (around 2~3 epochs, batch size=1)
3. 30 original image are quite similar and augmented training images makes the model wouldn't learn unexpected features such as specific shapes and directions.

3. result

I use first 25 images and their augmented image as training set, last 5 image as test set. The results after 1500 iteration of training are shown below:



using evaluation method Foreground-restricted Rand Scoring after border thinning V_rand and Foreground-restricted Information Theoretic Scoring after border thinning V_info:

ID	V rand	V info
25	0.37423595445261937	0.6003793298841478
26	0.3464636265094728	0.7024502530329145
27	0.26775999844411097	0.6168632849151061
28	0.1285957440617486	0.45878492486183153
29	0.0997353568053952	0.260261983407338

Reference:

1. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. (2015)
2. Çiçek, O., Abdulkadir, A., Lienkamp, S.S., Brox, T. and Ronneberger, O.: 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. (2016)
3. Simard, P.Y., Steinkraus, D. and Platt J.C.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. (2003)
4. Yu, F. and Koltun, V.: MULTI -SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS. (2016)
5. Wang, P.Q., Chen, P.F., Yuan, Y., Liu, D., Huang, Z.H., Hou, X.D., Cottrell, G.: Understanding Convolution for Semantic Segmentation. (2018)
6. Geoffrey Hinton on max pooling (reddit AMA), available: https://mirror2image.wordpress.com/2014/11/11/geoffrey-hinton-on-max-pooling-reddit-ama/
7. Receptive Field calculator, available: https://fomoro.com/research/article/receptive-field-calculator