Texas A&M University

Brief Draft Of New Network Architecture

Zheyong Sun

July 8, 2019

1 Overview

This is a brief summary of an idea about a network architecture proposed by Professor Dr.Jiang. Flood-Fill Network[1] was proposed in 2016 and applied in the segmentation of 3D images. This new architecture gets the idea of network iteration and Field of View(FOV) from FFN[1]. For simplicity, let's start from 2D image segmentation and FOV of it is 2D as well.

2 Start with 2D images

The overall structure of this new architecture is shown in Fig2.1. The generation of this new architecture is inspired by the ideas of Field Of View(FOV) and iteration used in FFN. We take the original image and FOVs produced based on an initial probability map as input for the first CNN. We also apply a mechanism similar to belief propagation to exchange the pixel value between two FOVs. The first CNN works in the iteration for training and its output has the same size as the input FOVs.

The output of the first CNN combined with the original image is fed into the second CNN. The purpose of the second CNN is to generate the probability map, the pixel value of which

represents the probability of being classified as boundary or target objects. If this probability map satisfies the demand then we treat it as the segmentation result. If it does not satisfies, we will produce FOVs based on it for the next iteration of the whole network. There are two iterations circle in this network, the small one outputs the proceed FOVs and the big one, which contains the first small circle, generates the final probability map.

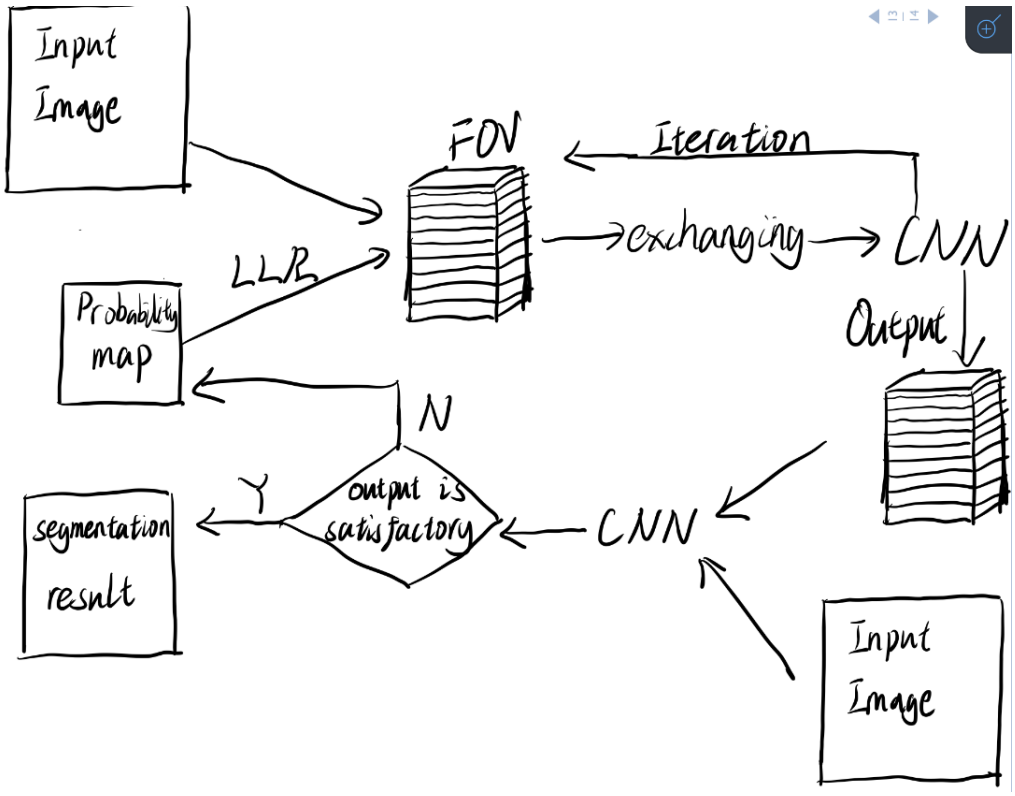

Figure 2.1: Overall structure of the network

2.1 Input data and FOV

In the first CNN, we fed the network with the original image combined with FOVs. For example, the size of the input image is 128*128 and the size of each FOV is 15*15, we will crop a 15*15 image from the original image, which has the same coordinate with that of FOV. Then we will stack these two 15*15 image together and such a operation is the same for each pixel in the original image. The final output of the first CNN is 128*128*15*15. It is easy to generalize it to the application of 3D images. We need to build Field of View(FOV)s for the first CNN first. FOV is similar to kernel or filter in CNN. We could run an already

existing segmentation algorithm to get an initial probability map. For those pixels on that probability map, we use Log-Likelihood Ratio to transfer them to generate FOVs. The function of Log-Likelihood Ratio is:

$$v = log_2\left(\frac{P}{1-P}\right) \tag{2.1}$$

The reason why we use Log-Likelihood Ratio to replace the 0-1 binary label is that the extended range from neg infinity to infinity providing more stability. When the 0-1 binary value closes to 0.5, which means that the network cannot judge this pixel should be cell or boundary, the LLR value of it is near to 0.

Assuming that the initial probability map is a 4*4 image and we build 3*3 FOV for each pixel on it. The setting of boundary pixels is the first thing we need to consider. We can pad the image to use the boundary pixels or ignore them. let' s just ignore them here, so we could get 4 FOVs marked with different colors on Fig2.2. The input of the first CNN is the stack of these 4 FOVs, A,B,C,D.

The first CNN we will introduce in the next section could convert each FOV to a primary probability map. The size of this primary probability map is the same with each corresponding FOV. These output primary probability maps will be treated as the input for the next iteration until the changing of one iteration below a threshold or meeting the number we set.

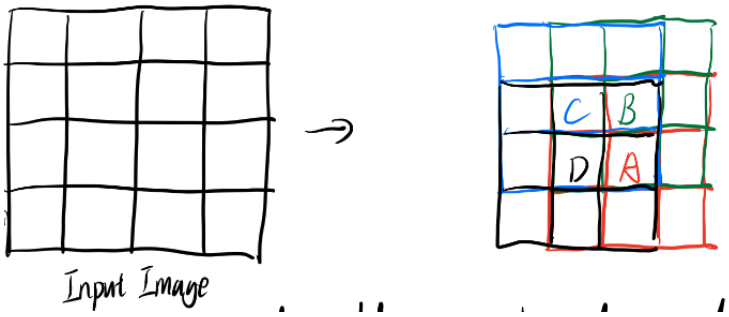

Figure 2.2: 4*4 input and its 4 kernels

2.2 The First CNN and Belief Propagation

In the iteration training process, we apply a new mechanism, which is similar to the idea of belief propagation in machine learning. We want each FOV to be able to communicate with other FOVs with the overlapping pixels because we think it may help against the misalignment and anisotropy in 3D EM images. The method of communication is exchanging

pixel values. We use kernel A and C in Fig2.2 as examples and show them in Fig2.3. In FOVs, each pixel only exchanges its value with another pixel in one other FOV. In Fig2.3, we could find that FOV A and C have their center pixel overlapping with each other. Then we switch the pixel in the FOV C overlapping with the center pixel in FOV A and the pixel in the FOV A overlapping with the center of C. The blank around C and A does not mean that the value of these pixels is zero, I just omit them for simplicity. In fact, these omitted pixels with be exchanged with other FOV not only just between A and C. For example, a 3*3 FOV like A and C will exchange its 8 out of 9 pixels with 8 other FOVs and the center pixel keeps its value forever.

Each input stack is composed by a box from original image and a box from probability from the same position and has the same size. We may training them one by one or using a small batch. After each epoch, we will executed the exchanging process and feed those stacks into the first CNN for the next iteration.

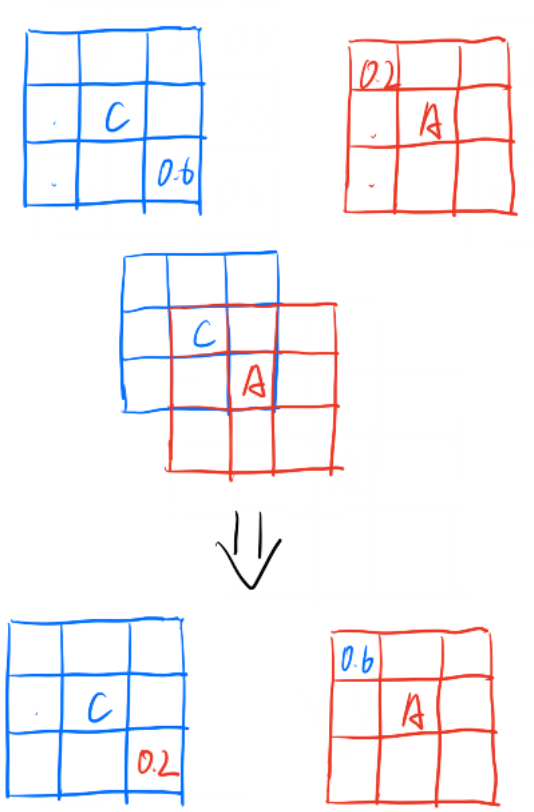

Figure 2.3: pixel value exchanging

2.3 The Second CNN and Big Iteration

When the output primary probability maps of each FOV are fixed, we will feed all of them combined with the original image into the 2nd CNN to merge them into a 2D boundary map with the same size of input 2D image as Fig2.1. We expect this boundary map can become the segmentation result eventually. The ground truth label for this period is the original full-sized annotated label. The output of the second CNN is the full-sized probability map but it may not performs well after the first round. We will generate the FOVs based on these segmentation maps and start the second round of the whole process, this the aforementioned Big Iteration. There are two kinds of iterations in the whole network, the first small iteration is to generate the FOVs and included in the second big iteration, which generates the final segmentation map. If the generated probability map fits our performance requirement, we will let is be the final segmentation result.
In such a network, the training times of the second CNN is much less then the first one, we need experiments to verify the actual performance.

2.4 Inference

In the inference or test process, there are something we need to figure out. Firstly, the input FOVs are produced based on another probability map, we could get it from other algorithms, this is the same with the training process. Then about the network, we still need iterations on both the first small circle and the whole network. The iteration will stop when the performance improvement is below a small threshold in an iteration, or when the number of iterations is above a big threshold.

3 extending this idea to 3D images

When we come to 3D images like SNEMI3D dataset, we need to adjust the setting aforementioned. Field Of View(FOV) turns from a square to a cube, but the length of the Z-axis of it should be tested in experiments. In 3D images, the size of the overall stack FOVs will be a huge problem due to the limited size of memory. We may not be able to initialize a FOV for each voxel and sampling may work.

Now we will use an example 3D image to go through this pipeline to show the dataflow. Given an original 3D image D with the size of 128*128*12 and we choose the size of FOV

to be 15*15*7. We apply another segmentation algorithm on D to get an initial probability map, then use formula2.1 on every voxel of that probability and crop a 15*15*7 cube on each voxel except the cube beyond the boundary. Each stack of input is 15*15*14, which is combined by two cube with the same size. The number of stack we have is $(128-15+1)*(128-15+1)*(12-7+1) = 114*114*6$. We could apply belief propagation before or after CNN. Each FOV with exchange $15*15*7-1$ pixels with other $15*15*7-1$ overlapping FOVs. We for CNN, we should use weight sharing so that the same filter is used for different FOVs. In each iteration, the size of the output is 114*114*6*15*15*7.
When the first iteration stop, the output FOVs with size of 114*114*6*15*15*7 in total combined with the 128*128*12 original image is fed into the second CNN. There is no belief propagation for the second CNN and the output is 128*128*12 probability map. We just go through this CNN for just once and check the output probability map whether it meets our requirement. If it is, then it will be our final segmentation result. If not, we will produce another FOVs with the size of 114*114*6*15*15*7 based on this probability map for the next round of the whole network.

References

[1] Michal Januszewski, Jeremy Maitin-Shepard, Peter Li, Jörgen Kornfeld, Winfried Denk, and Viren Jain. Flood-filling networks. *CoRR*, abs/1611.00421, 2016.