**Implementation of SegNet.**

**Acquiring images (CT system)**

104 rice samples were cultivated and for each rice sample 80 slices were reconstructed, so totally we have 8320 origin slices need to be segmented and analyzed.

**Challenges in the rice culm segmentation**

Our task here is to segment the rice culm in each slice. Disturbances such as leaf sheath adhering to the culm (Figure N1a), artifacts (Figure N1a and N1b), and contrast varying from different slices (Figure N1b) make segmentation of rice culm difficult. Therefore we expect a robust algorithm to segment the rice culms in different slices with various noise and of different contrast.

C:\Users\10214\Desktop\8-吴丹 20181103\N1.tif

Figure N1 Slices with various noise and of different contrast. (a) Leaf sheath adhering to the culm, (b) contrast varying from different slices, (a-b) artifacts.

**Solution Selection and technology roadmap**

Classical segmentation methods, such as threshold segmentation, didn’t perform well at distinguishing between rice culms and background. We tried to apply SegNet to slice segmentation, and good results had been achieved, IoU, Precision, Recall and F-measure (%) for the test set were 0.727, 0.749, 0.961 and 84.2% respectively. SegNet is a fully convolutional neural network architecture for semantic pixel-wise segmentation, which is appropriate for this task. For more details, see <http://mi.eng.cam.ac.uk/projects/segnet/tutorial.html>. The specific technology roadmap of using SegNet is shown as below (Figure N2):

C:\Users\10214\Desktop\8-吴丹 20181103\N2.tif

Figure N2. Technology roadmap of using SegNet

**Detailed methodological description**

**Part1: Off-line training**

**1) Manual segmentation using Photoshop software**

SegNet requires a dataset of input images with corresponding label images, in which each pixel was labeled with its class [1]. In training section, we randomly selected 200 slices from 8320 origin slices. These slices were manually segmented using Photoshop software to obtain ground truth labels. In each label, the foreground gray value of rice culms is set one and the background gray value is set zero. It took approximately 10-15 minutes to segment a slice with resolution of 1803 x 1803. One of the origin slices and its corresponding ground truth label is shown as follows:

**C:\Users\10214\Desktop\8-吴丹 20181103\N3.tif**

Figure N3. An origin slice and its corresponding label image

**2) Dataset building and data augmentation**

The next step is to build training set, validation set and test set. We divided the dataset into three parts: 160 slices for training set, 20 for validation set and 20 for test set. To enlarge the training set, for each slice in training set, we rotated it and adjusted its contrast to generate seven new training samples.

C:\Users\10214\Desktop\8-吴丹 20181103\N4.tif

Figure N4. Data Augmentation by rotation and contrast adjustment

The contrast adjustment function is:

For contrast--:

(1)

For contrast-:

(2)

For example, there are two steps from figure (N4a) to figure (N4f):

i) Rotate the origin slice by 90 degrees;

ii) Gray value of each pixel in the rotated slice was multiplied by 0.75 and then added 32.

After data augmentation, we have 1280 slices in training set.

**3) Clip the slices to meet the input requirement of SegNet**

The input of SegNet is an image of resolution 480 x 360, and the output of it is an image of the same size with each pixel labelled with its class. Since the size of each origin slice is 1803 x 1803, which didn’t meet the input requirement of SegNet, we enlarge and clipped each slice into 24 patches by following steps:

i) Expand each origin slice and its corresponding label to resolution 1920 x 2160 so that it can be exactly divided into 24 patches size of 480 x 360 with no remainders. Supplementary pixels at the right edge and bottom edge are set zero.

ii) Clip each expanded image and corresponding label into 24 patches (480 x 360).

C:\Users\10214\Desktop\8-吴丹 20181103\N5.tif

Figure N5. Enlarge and clip an origin slice into 24 patches

The corresponding code is expressed in C++ language as follows:



**4) Reduce patches with no pixels belong to foreground**

In training set, there are totally 30720 (1280 x 24) patches now. The number of background pixels in each slice is far larger than that of foreground pixels. Taken the balance of positive and negative pixels into consideration, we divided these patches into two parts:

i) 9973 patches with both foreground and background pixels;

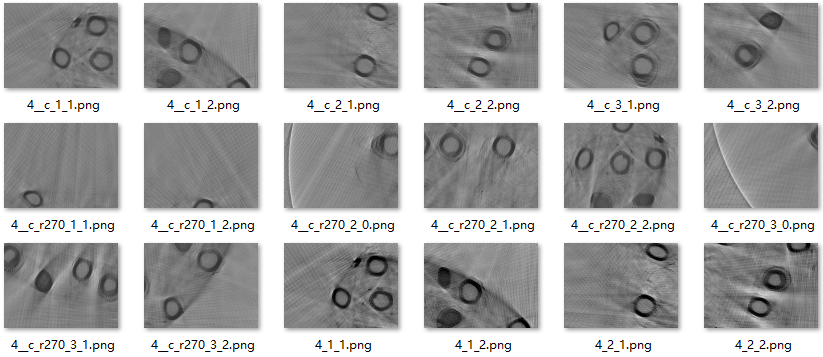


Figure N6. Patches with both foreground and background pixels

ii) 20547 patches with only background pixels.

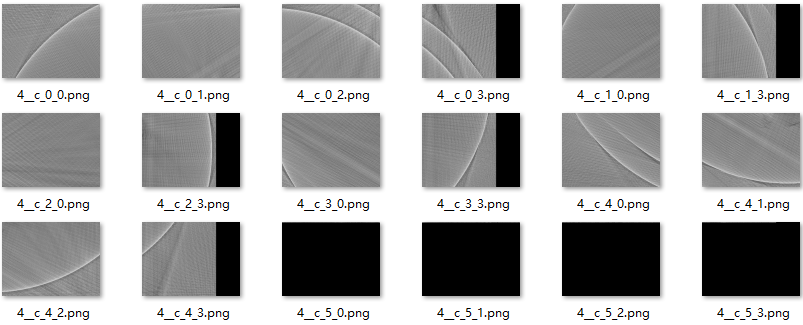


Figure N7. Patches with only background pixels

All 9973 patches with both foreground and background pixels remained as training samples, another 200 patches were randomly chosen as training samples from 20547 patches with only background pixels. So finally, the training set is composed of 10173 samples.

**5) Training SegNet** (tutorial: <http://mi.eng.cam.ac.uk/projects/segnet/tutorial.html>)

The SegNet training was implemented on Ubuntu system.

According to SegNet tutorial, we arrange the folders as follows:

The whole dataset including 10173 training, 480 validation (20 slices×24 patches) and 480 test samples (20 slices×24 patches) were all stored in several subfolders of the “riceCulm” folder.

**Dataset file structure**

riceClum/

val/ # 480 patches from 20 origin validation samples

valannot/ # 480 patches from 20 ground truth validation labels

train/ # 10173 patches from 160 origin training samples

trainannot/ # 10173 patches from 160 ground truth training labels

test/ # 480 patches from 20 origin test samples

testannot/ #480 patches from 20 origin test labels

test.txt

train.txt

val.txt

In “train.txt” and “val.txt” stored the absolute path to training samples and the absolute path to validation samples respectively.



**Model file structure** (all model files can be download from <https://github.com/alexgkendall/SegNet-Tutorial>)

Model\_riceCulm/

Inference/

log/

Training/

compute\_bn\_statistics.sh

segnet\_riceCulm\_deploy.prototxt

segnet\_riceCulm\_solver.prototxt

segnet\_riceCulm\_train.prototxt

train\_riceCulm\_segnet.sh

Some documents have been renamed and some parameters have been adjusted as required. Changes in the model files are listed as follows:

“segnet\_riceCulm\_train.prototxt”

The training data source and validation data source were changed to the absolute path to our data file. Considering the computing capability of GPU, we change the batch size of training process to 4 and the batch size of validation process to 2.



Though we have removed a number of patches with only background pixels in training set, this was not enough to ensure the balance between positive and negative samples. The number of pixels in background was still far larger than that in foreground in the training set. In this case, the network would incline to determine a pixel as a background pixel. So the network adopt the class\_weighting parameters, with which the loss would increase more if the pixels of minor class were incorrectly discriminated, to reduce such tendency.

Calculate class\_weighting for each class:

(3)

Where weight0 is the weight of background and weight1 is the weight of foreground; , means the total number of pixels that belong to foreground and background in all training samples respectively. According to formula 3, weight0 is 0.514825, weight1 is 17.3639 and we simply considered that the weights of validation samples are the same as that of training samples.



“segnet\_riceCulm\_solver.prototxt”



Line 1: net: "/home/xiongxiong/SegNet/Model\_riceCulm/segnet\_riceCulm\_train.prototxt" (the address of the network)

Line 3 test\_iter: 240

The total number of samples in the validation set is 480. These samples were divided into batches to execute and the number of samples in each batch is batch\_size (equal to 2). We set test\_iter 240, since after 240 iterations we could complete all 480 validation samples.

Line 4: test\_iterval: 2544

The batch\_size for the training process was set as 4, it would take 2544 iterations to complete all 10173 samples in the training set. So, test\_iterval was 2544, which meant after 2544 iterations when all training samples have been performed, a test would be done.

Line 5-8: base\_lr, the base learning rate, was set 0.001. In the iterative process, the learning rate would be adjusted by Ir\_policy, “step” method. Stepsize represents changing the learning rate after “stepsize” iterations per training session and the value of “gamma” means when updating learning rate, the new learning rate will be gamma by current learning rate. Here, the stepsize is set as 150000000, which is larger than max\_iterval, this meant that the learning rate would not be updated during the whole training process.

Line 9: 20 times training, displayed on screen once.

Line 10: The maximum number of iterations is 25440.

Line 12: Every 2544 iterations, the training model and current state will be saved.

Line 13: The path to save the models.

Line 14: Train the model on GPU

Here, fine-turning is applied by using “VGG\_ILSVRC\_16\_layers.caffemodel”. By providing the weights argument to the Caffe train command, the training will begin with a pretrained encoder.

train\_riceCulm\_segnet.sh



After 25440 iterations, we obtain the final rice culm SegNet training model (“segnet\_riceCulm\_iter\_25440.caffemodel”).

**6) Performance evaluation of the testing set**

Before we applied the model to segment all 8320 origin slices, we evaluate the performance of the trained model on the test set, which including 480 patches from 20 origin slices.

Compute BN statistics for SegNet

Open the script “compute\_bn\_statistics.sh” and change the training weight file to the latest saved model (“segnet\_riceCulm\_iter\_25440.caffemodel”).



The testing process is as follows:

i) Run the model to segment 480 patches.

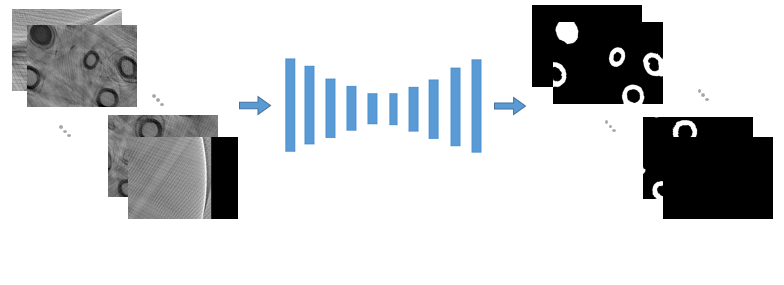


Figure N8. Applying the trained model

ii) Patches that belong to the same origin slice were merge into one binary image with resolution of 1920×2160 according to the their positions with respect to the origin slice, then the merged image was clipped into image with resolution of 1803 x 1803.

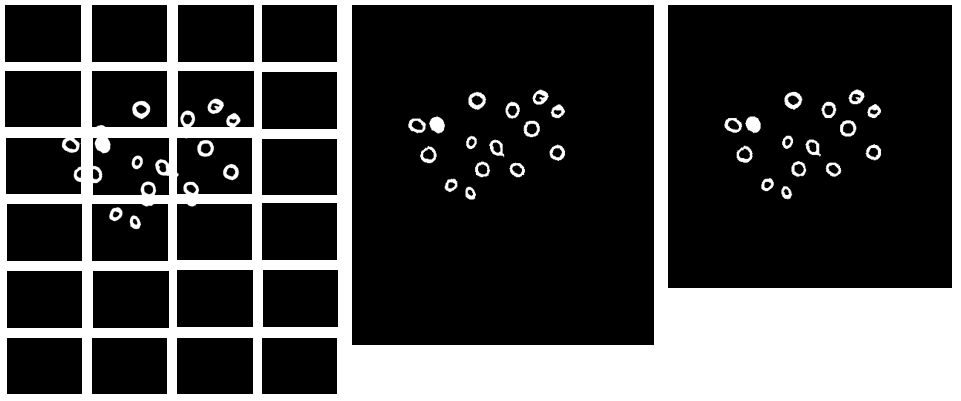


Figure N9. From patches to segmented image

iii) Eroding operation was implemented and small regions were removed to obtain the final segmented result.

To evaluate the performance of the segmentation, we adopted four indicators, including IoU, Precision, Recall and F-measure [2]. The computational formulas for them are provided in Eqs. 4-7.

(4)

(5)

(6)

(7)

Where TP, TN, FP and FN represent the numbers of true positives, true negatives, false positives, and false negatives, respectively. True positives are when the predicted results and ground truth are both rice culm pixels. True negatives are when the predicted results and ground truth are both background pixels. False positives are when the pixels classified as rice culm pixels but which actually belong to the background. False negatives are the pixels that belong to rice culms but not correctly classified. The corresponding code for calculate IoU, Precision, Recall, and F-measure is expressed in C++ language as:



For 20 origin slices in the test set, IoU, Precision, Recall and F-measure(%) were 0.727, 0.749, 0.961 and 84.2% respectively.

**Part2: On-line segmentation**

After the SegNet model for rice culm segmentation has been trained well, we can apply it to segment all 8320 origin slices. This contains mainly three steps:

1) Clip each slice into 24 patches using the same method mentioned before;

2) Pixel-wise segmentation using pre-trained model;

3) Merge the patches that belong to the same slice and clip it into size of 1803 x 1803;

4) Erode operation with kernel size 3 x 3 on each cropped image;

5) Obtain the finial segmented slices.

**Reference**

[1] Vijay B, Alex K, Roberto C. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, PP(99):1-1.

[2] Xiong X, Lingfeng D, Lingbo L. Panicle-Seg: a robust image segmentation method for rice panicles in the field based on deep learning and superpixel optimization. Plant Methods, 2017, 13(1):104.

[3] Gary B, Adrian K. Learning OpenCV: Computer Vision in C++ with the OpenCV Library. Apress 2013.