SEGMENTATION OF CELL PICTURE FOR CELLARI

Liguang He(s190008), Xiaohan Lyu(s182354), Jingshi Hu(s181387)

DTU Electrical Engineering

ABSTRACT

Cooperated with Cellari who is focus on biological images segmentation, we present a practical solution on gland cell segmentation with dataset from Gland Segmentation in Colon Histology Images (GlaS) by using convolutional neural network for semantic pixel-wise segmentation named SegNet. Except solving gland cell segmentation problem we compared performance between Full SegNet which is based on VGG16 and a smaller version of SegNet called SegNet Basic on F1 score. The quantitative assessments reveal that SegNet Basic provides better performance and competitive efficiency in terms of training time on this dataset. The Pytorch implementation of the project is provided on [1]

***Index Terms*—** Cell segmentation, SegNet, Data Augmentation

1. INTRODUCTION

In this project, we cooperate with company Cellari, who provides state of the art artificial intelligence models for bioimage segmentation in combination with an easy and intuitive user interface. Cellari enables researchers and non specialists to take full advantage of recent developments in artificial intelligence. [2]

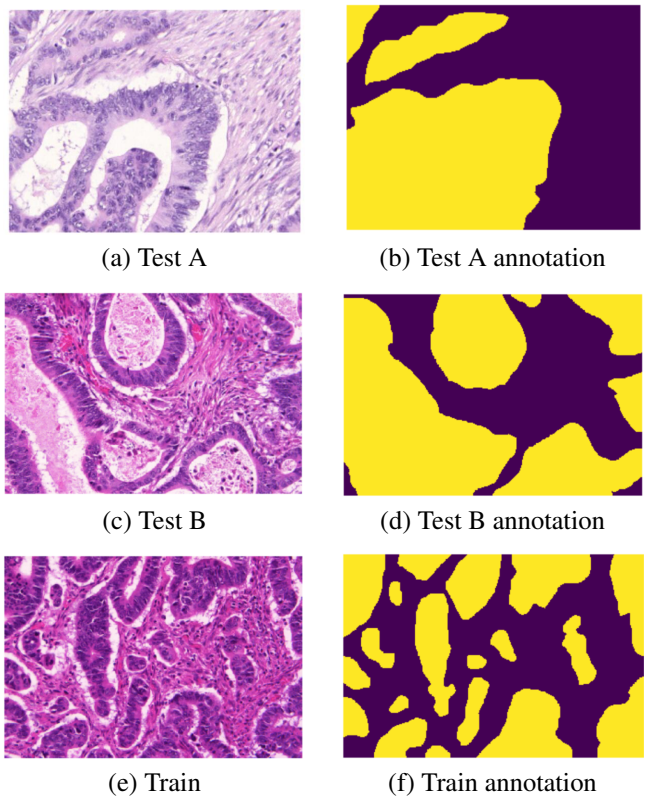
The main task of our project focuses on gland cell segmentation that dataset comes from Gland Segmentation in Colon Histology Images (GlaS) challenge.[3]

Our main contribution is to analyze performance of two types of SegNet networks and choose the appropriate network for the dataset. First, data augmentation is applied to enlarge the dataset to provide enough samples for the neural network model that is described in section 2. Then, two network architectures are introduced in secntion 3. One of the networks is called SegNet Basic and the other one is called Full SegNet. In following we analyze and compare the performances of the SegNet Basic and Full SegNet in section 4. Finally, the conclusions and future improvements are given.

2. DATASET AND DATA AUGMENTATION

Our dataset comes from Colon Histology Images Challenge Contest (GlaS) held at MICCAI 2015. [3] It consists of 165 images derived from 16 H&E stained histological sections of stage $T3$ or $T4$ colorectal adenocarcinoma and 165 corresponding manual annotation images. A pathologist delineated the boundary of each individual glandular object on that visual field.So we can use this manual annotation as ground truth for automatic segmentation.

The entire dataset is divided into **Training Part**, **Test Part A**, and **Test Part B**. In our project, the Training Part is used as train dataset, Test Part A as test dataset and Test Part B is for F1 score calculation. The example images of the three parts can be seen in Figure 1.



(a) Test A (b) Test A annotation

(c) Test B (d) Test B annotation

(e) Train (f) Train annotation

Fig. 1. Representative example images of the three parts.

The total number of raw training images are 85 which it is believed they are too small for network training. Because the training quality of neural network largely depends on the scale

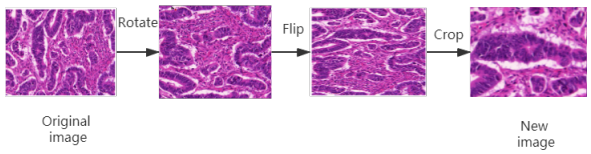


Fig. 2. The result of data augmentation

and quality of the raw data. Normally the larger scale and better quality of raw data, the stronger generalized capability of the model. For this reason, we need to implement data augmentation to expand the size of the samples.

In our case the random rotation, flipping and cropping are applied on each batch to ensure that the pictures are different in each epoch. The way of data augmentation in our case is shown below.

1) Choose one image and the its corresponding annotation.

2) Generate a random angle to rotate the image.

3) A 50% chance to flip the image horizontally, and a 50% chance to flip the image vertically.

4) A 50% chance to crop the image to a fixed size after zooming in, and a 50% chance to crop the image to a fixed after reducing the image and padding.

5) Convert the image to a tensor and normalize its pixel value from 0 to 1.

The result of data augmentation are shown in Figure 2. Using this way of data augmentation, we can make sure that each set of photos entered into the network is different.

3. NETWORK ARCHITECTURE

The method we adopted is based on semantic segmentation which is a classification on pixel-wise. In other words semantic segmentation understands images at pixel-wise. In our project the segmentation can be seen as binary classification. The network model is aim to distinguish whether the pixel is background or cell. The method we used is encoder-decoder architecture. This is based on FCN (Fully Convolutional Network) by using pooling to decrease spatial dimension (encoder) and decoder gradually recovers the spatial dimensions and details.

SegNet is composed by encoder and its decoder network, followed by a final softmax layer. The encoder network contains convolution layers, batch normalization layer, ReLu layer and pooling layer [4]. Each convolution layer is followed by a ReLu and max pooling with a 3×3 window and stride 2.

The encoder is aimed to produce a set of feature maps by using convolution as a filter. The batch normalization is applied after each convolution layer to accelerate convergence of model [5] [6]. Batch normalization works differently in training and test. In training part, batch normalization layer normalizes the distribution of layer inputs (the feature map which after convolution) in forward propagation [3]. But during test, the network is not expected to learn anymore so each batch normalization layer uses mean and variance that already stored during the training process. Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). By using this normalization method, the input of activation layer can be more linear to make activation function works more effectively.In pytorch we can set different work method by setting model condition (training or evaluation).[7]

ReLu function comes after batch normalization. ReLu function is shown in Figure 3.

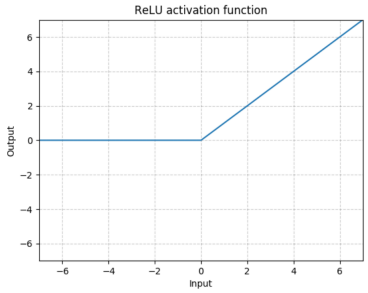


Fig. 3. ReLu[8]

The output of ReLu function is:

$$ReLu(x) = max(0, x) \tag{1}$$

Then max pooling is used to decrease image size. We can get a 1×1 feature point from a 2×2 feature area by using 2×2 pooling. In encoder network, the position that 1×1 feature point in 2×2 feature area will be recorded, that called pooling indices.

The decoder upsamples the input feature map to produce the sparse feature maps by using pooling indices, like Figure 4.

Then using deconvolution to fill the missing content. After that a batch normalization is used to each feature map. The softmax layer is in the last of whole decoder process that provides the maximum probability of each pixel in its corresponding class.

Regarding to the full SegNet architecture, it is based on VGG 16 network but disconnecting the last 3 full connected layers so the encoders of it consists of 13 convolution layers and hence the decoders contains 13 layers [4]. The architeture is described in Figure 5.

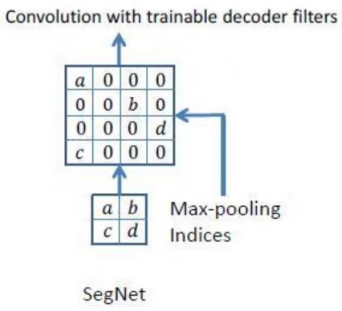


Fig. 4. unsampling

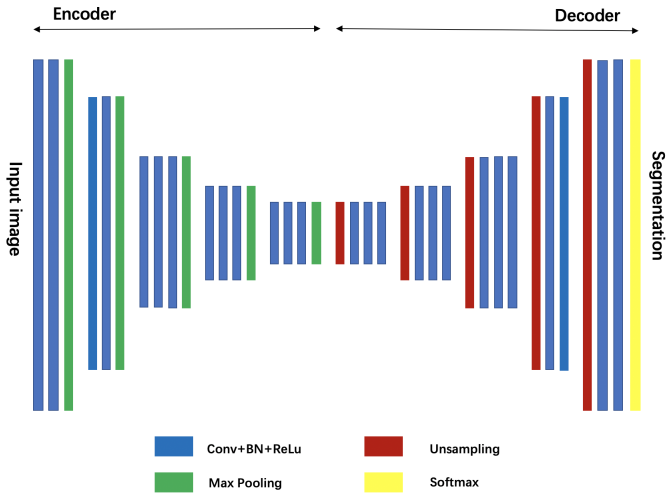


Fig. 5. Full SegNet

The SegNet Basic is a smaller version which has 4 encoders and 4 decoders only [4]. Its architecture shows as Figure 6. Similar to the full SegNet, Batch normalization and non linearity ReLu are present after each convolution layer.

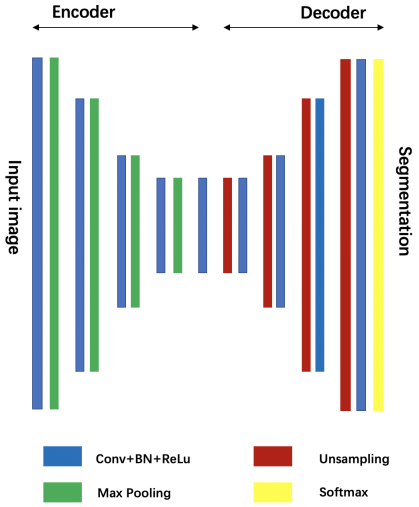


Fig. 6. SegNet Basic

4. TRAINING AND ANALYSIS

The weights of encoder and decoder are initialized by He initialization [9]. Adam with fixed learning rate and betas between 0.9 and 0.999 are used to seek optimal values. The loss function uses BCE loss (Binary Cross Entropy) that is calculated as:[10]

$$BCE\ Loss = y \cdot log(x) + (1 - y) \cdot log(1 - x) \qquad (2)$$

Where y is target, x is prediction.

To verify the performance of full SegNet and SegNet basic we use direct intuitive comparison between prediction and annotation and F1 score, which is defined as

$$F1\ score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

where

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}$$

TP: the number of true positives;
FP: the number of false positives;
FN: the number of false negatives from all images;

Then the following parameters are tuned to find the highest performance on the test data set:

1) At the beginning the learning rate was set to very large as 10, then reduce it to 0.1, 0.01, 0.005, 0.001, 0.0001, 0.0005. For each learning rate we observed the loss curve and calculate F1 score and choose the one that loss curve decreases stably. The higher learning rate makes loss curve oscillate around lowest loss value while the lower learning rate will lower convergence speed of loss curve.

2) Regarding to epoch number, the epoch number is set to increasing and the training is stop when the change of loss value is very slow.

3) The batch size direct influence the usage of RAM. If it is set to large the RAM is easily overused and program is crash down. The target is to pick up a batch size to ensure loss converge fast enough and stable.

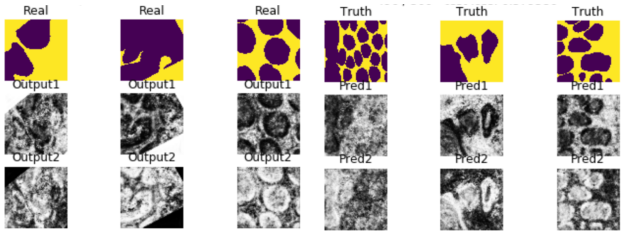
4) L2 Regularization is used to prevent overfitting. It is specified as weight decay in Pytorch.

The process to choose combination of parameters to reach optimal performance takes quite a long time and finally the parameters chosen is shown in table 1.

Table 1. Net Parameter

net name	SegNet Basic	Full SegNet
epoch number	400	500
batch size	10	10
learning rate	5e-4	5e-4
weight decay	1e-4	1e-4

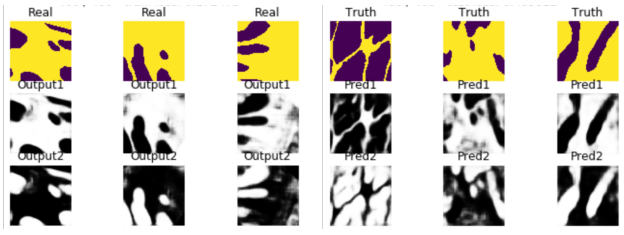
Figure 7 and Figure 8 show the intuition result where we can see that the SegNet Basic is able to clearly distinguish two classes from original image while Full SegNet is more focus on feature details. In our case the Full SegNet can capture more details of cells but its performance become poor if cells are close to each other.



Train prediction Test prediction

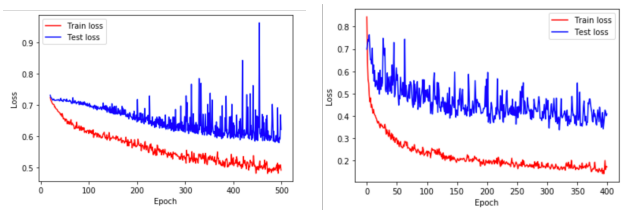
Fig. 7. Full SegNet Results

This is also verified on the F1 score. For the SegNet Basic the overall F1 socre can achieve 0.8535 but the F1 score of Full SegNet is 0.6590 only. Figure 9 indicates the train and test loss in both Full SegNet and SegNet Basic.



Train prediction Test prediction

Fig. 8. SegNet Basic Results



Full SegNet SegNet Basic

Fig. 9. Loss curve

Theoretically deeper network has stronger capability than shallow network and it should perform better but this is not true in our case. The reasons we investigated are:

1) The size of training data set is too small which is 85 images only. Though data augmentation is applied many pictures are still similar therefore they do not help to improve training performance in Full SegNet.

2) Obviously there are more convolution layers in the Full SegNet that probably leads to feature information loss. Figure 10 and 11 show the last layer of convolution results in full SegNet and SegNet basic. It can be seen that more information are lost for Full SegNet when compared with SegNet Basic.

5. SUMMARY AND FUTURE IMPROVEMENT

5.1. Summary

According to the result of training and analysis we have solved our project's problem and successfully trained the model to distinguish cell from its background. Besides, during selecting network architecture we compared the SegNet Basic and Full SegNet and found SegNet Basic performs better than Full SegNet on this dataset though we believe the Full SegNet's performance will be improved if the dataset is large enough.

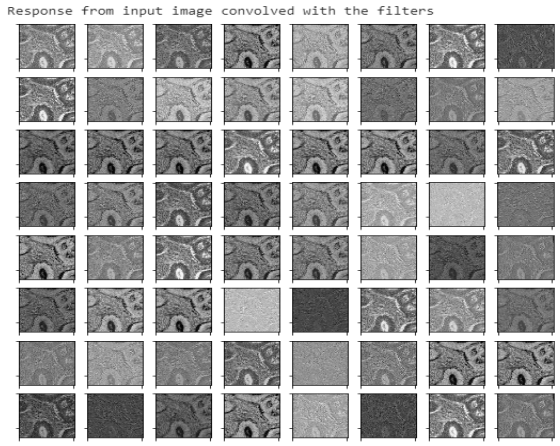


Fig. 10. SegNet Basic Information Tracking

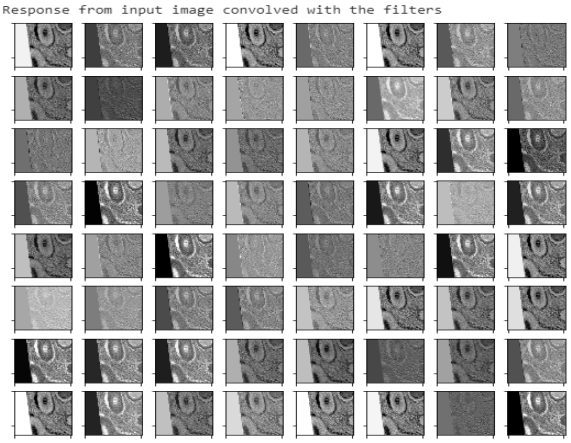


Fig. 11. Full SegNet Information Tracking

5.2. Future improvement

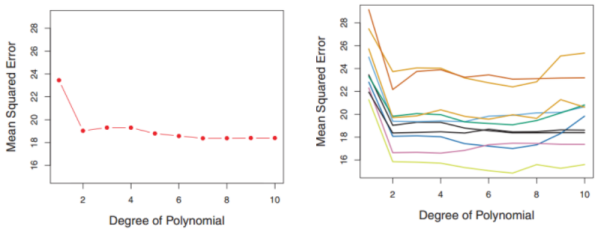
Although we solved the main problem of the project there are still some future improvements can be done to enhance network performance.

5.2.1. Cross-validation

The dataset was divided into three sub-dataset: training, test A, test B. We train the network model directly using train dataset and observe the loss value at test A dataset, then apply the network model directly on test B to calculate F1 score.

The first drawback of using dataset in this way is that the selection of parameters (e.g. learning rate, batch size) mostly depend on the pre-defined dataset. As Figure 12 shows the loss value under one way and ten different ways to divide dataset.

From this figure it can be seen that loss value varies under different methods and the corresponding degree on x-axle is also different. So if for example the training dataset is good



(a) One way (b) Ten different way

Fig. 12. Test result under different way to divide data set[11]

engouh but the test data is filled with noise that will lead to we cannot get optimal parameters for highest performance.

Second drawback is that if training dataset is used for trainning network model only, this means that only part of dataset are used for trainning. As we known, the more dataset the better performance but in the way we use the data does not fully utilize given dataset. As a result of it the performance of network model is influenced.

Above problems can be solved via cross-validation. There are several method about cross-validation, such as Leave-one-out cross-validation, K-fold Cross validation. A possible way to improve our project is to use k-fold cross validation. The main idea is that divide dataset into number of k, then random set one sub-dataset as test dataset (not put back) and train network model with rest sub-dataset. Repeat above procedure in k times and finally compute the average of loss value:

$$Loss = \frac{1}{k}\sum_{i=1}^{k} loss_i \tag{3}$$

5.2.2. Data Augmentation

Another way to improve performance of network is that use another data augmentation methods. Currently data augmentation is done by using affine transformation and crop. Other data augmentation ways like translation, scale, constant (inserts a constant into image unknown area) and reflection are believed to improve model performance.

Also, different normalization method can be used to reduced the noise of the loss curve. One way is to normalize image with mean and standard deviation of image. Firstly compute the mean and standard deviation of each channel of images, then using mean and standard deviation to normalize corresponding channel of images.[12]

6. REFERENCES

[1] "Code implementation on github," Website, https://github.com/xiaohanlyu/Deep-learning-project/tree/master/Hand%20In.

[2] "Home of cellari," Website, https://cellari.io/.

[3] Korsuk Sirinukunwattana, Josien PW Pluim, Hao Chen, Xiaojuan Qi, Pheng-Ann Heng, Yun Bo Guo, Li Yang Wang, Bogdan J Matuszewski, Elia Bruni, Urko Sanchez, et al., "Gland segmentation in colon histology images: The glas challenge contest," *Medical image analysis*, vol. 35, pp. 489–502, 2017.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[5] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry, "How does batch normalization help optimization?," in *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

[6] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[7] "Batchnorm2d," Website, https://pytorch.org/docs/stable/nn.html?highlight=batchnorm#torch.nn.BatchNorm2d.

[8] "Relu," Website, https://pytorch.org/docs/stable/nn.html?highlight=relu#torch.nn.ReLU.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[10] "Bceloss," Website, https://pytorch.org/docs/stable/nn.html?highlight=bce#torch.nn.BCELoss.

[11] "Cross-validation," Website, https://zhuanlan.zhihu.com/p/24825503?refer=rdatamining.

[12] "normalize," Website, https://pytorch.org/docs/stable/torchvision/transforms.html?highlight=normali#torchvision.transforms.Normalize.