Assignment-2 of Deep Learning in Computer Vision
Nuclei Segmentation

Xiao Hu[1], Boya Wu[2], and Wenjing Zhao[3]

1. xiahaa@space.dtu.dk, 2. s170061@student.dtu.dk, 3. s191118@student.dtu.dk

1 Nuclei Segmentation

We have investigated a instance segmentation problem by segmenting individual nucleis from original images. The Kaggle 2018 Data Science Bowl challenge dataset[1] is used for training (80%) and validation (20%). The following tasks have been done:

- Design an U-Net [1] architecture for segmentation.
- Evaluate performance in terms of mean average precision.
- Improve generalization by data augmentation, input normalization.
- Optimize the U-Net with different loss functions and evaluate their performance.

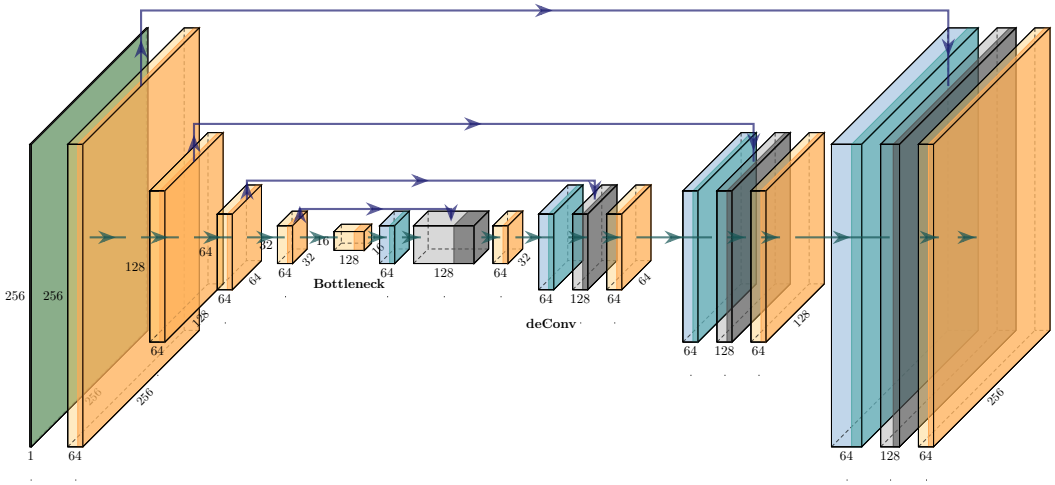

Fig. 1. U-Net architecture used in this assignment.

1.1 Network Design

We design a U-Net-like architecture which is shown in Figure 1:

- Use convolution with stride= 2 for down-sampling and deconvolution for up-sampling. We also tried the combination of Max-Pooling and up-sampling. The experimental results show similar performance. The prior option is favored since convolution with stride= 2 consumes less computational power than convolution with stride= 1 and then pooling.
- Convolutions are done with the kernel size being 3×3 and pad being 1. Deconvolution are processed with the kernel size being 2×2 and pad being 1.
- Dropout units are added before and after the bottleneck module.

We tried to implement the Mask R-CNN. However, the non-maximum suppression package doesn't match with the Pytorch version we are using, which make the compilation failure.

[1] https://www.kaggle.com/c/data-science-bowl-2018/data/

1.2 Data Preprocessing

Data Loading The dataset contains images of different size, which is not suitable for training. Two possible solutions could be use:

- Similar to [1], we could pad image with constant value or with neighboring pixels for training. In this case, when testing, a sliding procedure is needed to process larger image.
- Resize all images to a constant size.

Generally speaking, the first option is better than the second option since it doesn't loss any spatial information. However, it needs more workload. Because of the time limitation, we use the second option by resizing all images to 256×256.

Preprocessing We convert image to gray scale image since it demonstates better performance, see Figure 2. Besides gray scaling, we also tried several contrast enhancements since we found out that some images are low contrast. The results are shown in Figure 3. The improvements are slight, thus finally we use simple gray scaling without other image enhancement. After gray images are

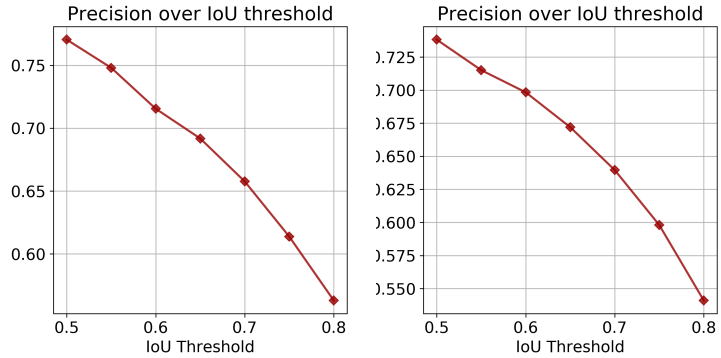

Fig. 2. Mean Average Precision over IoU threshold results: gray scale (left), RGB (right).

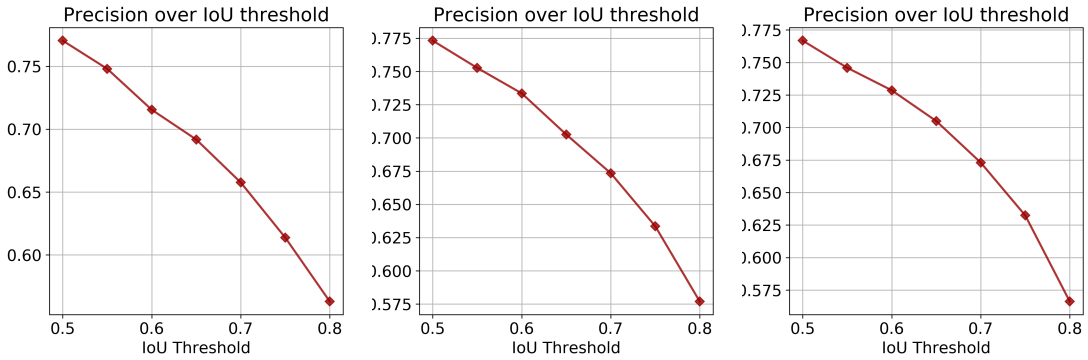

Fig. 3. Mean Average Precision over IoU threshold results: Gray scale (left), Gray scale+logrithm correction (middle), and Gray scale + sigmoid correction (right).

obtained, we do input normalization to normalize the input to zero mean and standard deviation of 1.

1.3 Post Processing

After obtaining the predict feature map from U-Net, the sigmoid function is applied to get the probability map. By thresholding and extract component connected using watershed algorithm, all predicted nucleis can be indentified.
For evaluation, we do as follows:

1. For all true masks, we use & binary operation to find its matched objects in the predicted map. If no match is found, then we add 1 to the false negative.
2. For matched objects, we select the one with biggest area and discard others.
3. Compute IoU using true mask and matched mask by & and | binary operation.
4. If IoU is over a threshold, then add 1 to true positive and mask out matched mask in predicted map.
5. After traversing all truth masks, false positive is counted as the remaining masks in the predicted map.

1.4 Further Improvement

Data Augmentation Data augmentation for this task is more complex than the previous assignment since both image and mask need to be augmented. In order to support data augmentation, we divide data augmentation as **image transformation** and **geometric transformation**. Image transformation is only applied to raw image including constrast jetter, etc. Geometric transformation includes affine transformation (no shearing), resize, crop, etc. Detailed augmentation is shown as follows:

- Affine transformation: rotation $(+/-\ 180)$, translation $(+/-\ 0.01 \times width, height)$, scaling $(0.8 - 1.2)$.
- RandomHorizontalFlip: flip in the horizontal way randomly (p=0.5).
- RandomVerticalFlip: flip in the vertical way randomly (p=0.5).
- Resize, rotate and center crop: resize to 350×350, randomly rotate and then center crop the image to 256×256.

We finally took random horizontal and vertical flip as data augmentation.

Erosion/Dilation After checking those worst segmentation images, we found out the major reason that the mean precision could not improve is because adjacent nucleis are segmented as a joint, which is even worse when large number of small nucleis are segmented as a big joint. In order to improve this case, we add erosion on the training masks to artificially segment close nucleis, while dilation is added at the end to compensate the side effect (thinner objects) of erosion. The gives us the best result on the validation data, as shown in the Figure 4.

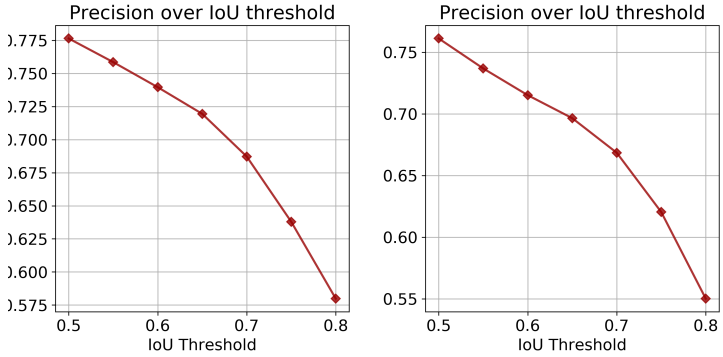

Fig. 4. Mean Average Precision over IoU threshold results with erosion/dialation using square mask: mask size (3×3) (left), mask size (5×5) (right).

Use Contour We tried to add contour as the third class, which aims to better seperate close nucleis:

1. Find contour on the training masks and label those pixels as border class.
2. Rewrite loss function using softmax since we are dealing with three class classification problem.
3. When doing instance segmentation, firstly treate border pixels as foreground pixels and segment. After that, we check if a border pixel is inside a segment (check if 4-neighbors are all masked pixels). We then delete those border pixels inside the segment.

Since border pixels inside an image is much less than other pixels, for class balance, we create weights using the inverse frequency of each class as:

$$weight_i = \frac{\sum_{N,H,W} 1}{\sum_{N,H,W} l == i}, \quad i = 1,\ 2,\ 3$$

Weights will further be normalized with their sum being 1.
Because of time limitation, we haven't thoroughly completed this improvement (step 3). We tried to process with the intermediate results, but the performance was not very promissing.

1.5 Result & Analysis

The mean average precision over IoU threshold is shown in Figure 4. It drops from $\approx 77\%$ when IoU threshold is 0.5 to $\approx 59\%$ when IoU is 0.8. Some segmentation results are shown in Figure 5. It can be seen that for nucleis with sufficient margin, the network can segment them very well. However, the real challenging part is the case where large number of nucleis are close to each other.

Figure 1.4 shows the result on stage 2 test images provided by the instructor. It can be seen input normalization severely change the appearance of the input images and make the segmentation worse than that without input normalization. It can be concluded that input normalization should be done in a proper way with regarding to the data at hand.

Several points need to be considered for future improvement:

- Use advanced region proposal network like the one in Faster R-CNN for better segmenting close objects.
- Proper way of doing data augmentation and input normalization for a specific task.

References

1. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)

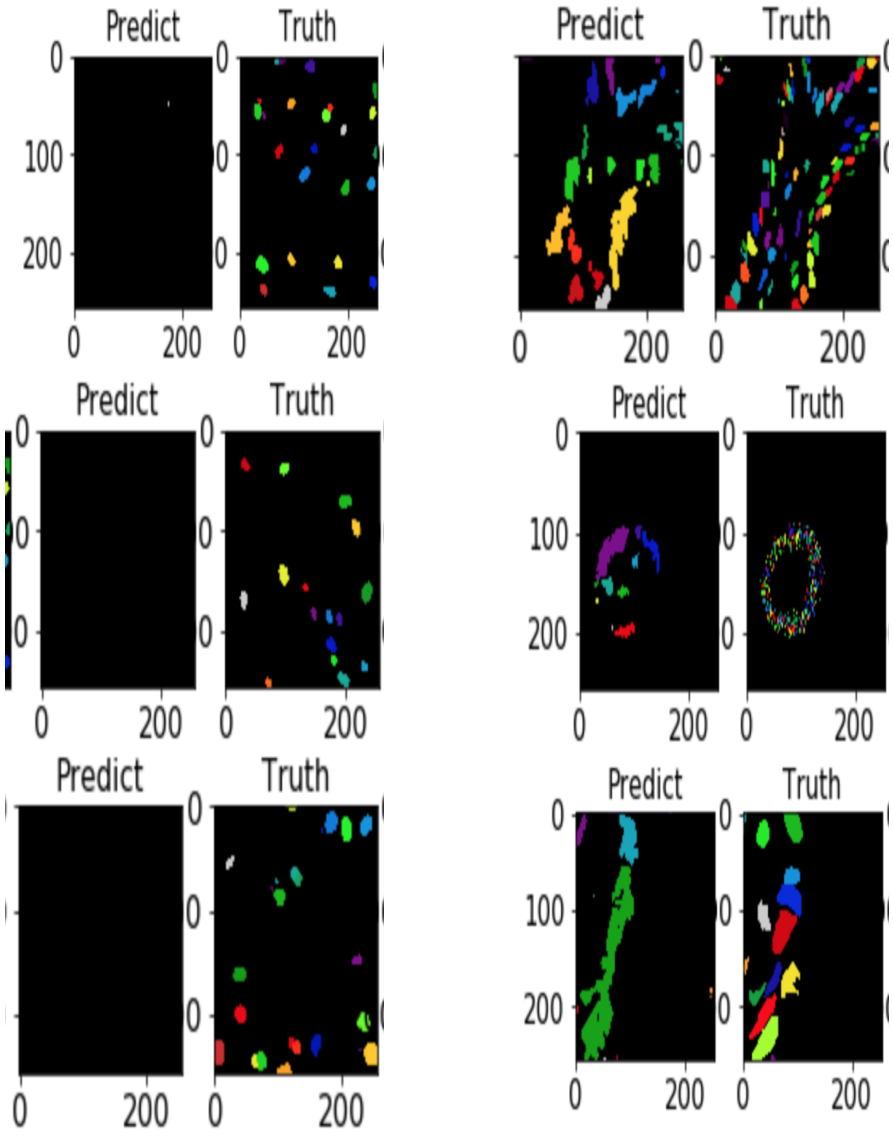

Fig. 5. Segmentation results: good (left), bad (right). Illustration of figures: since we will remove masks in the predicted map if a corresponding unique nuclei in true mask can be found, so for good results, you will see the predicted map as a blank image (all masks have been removed).

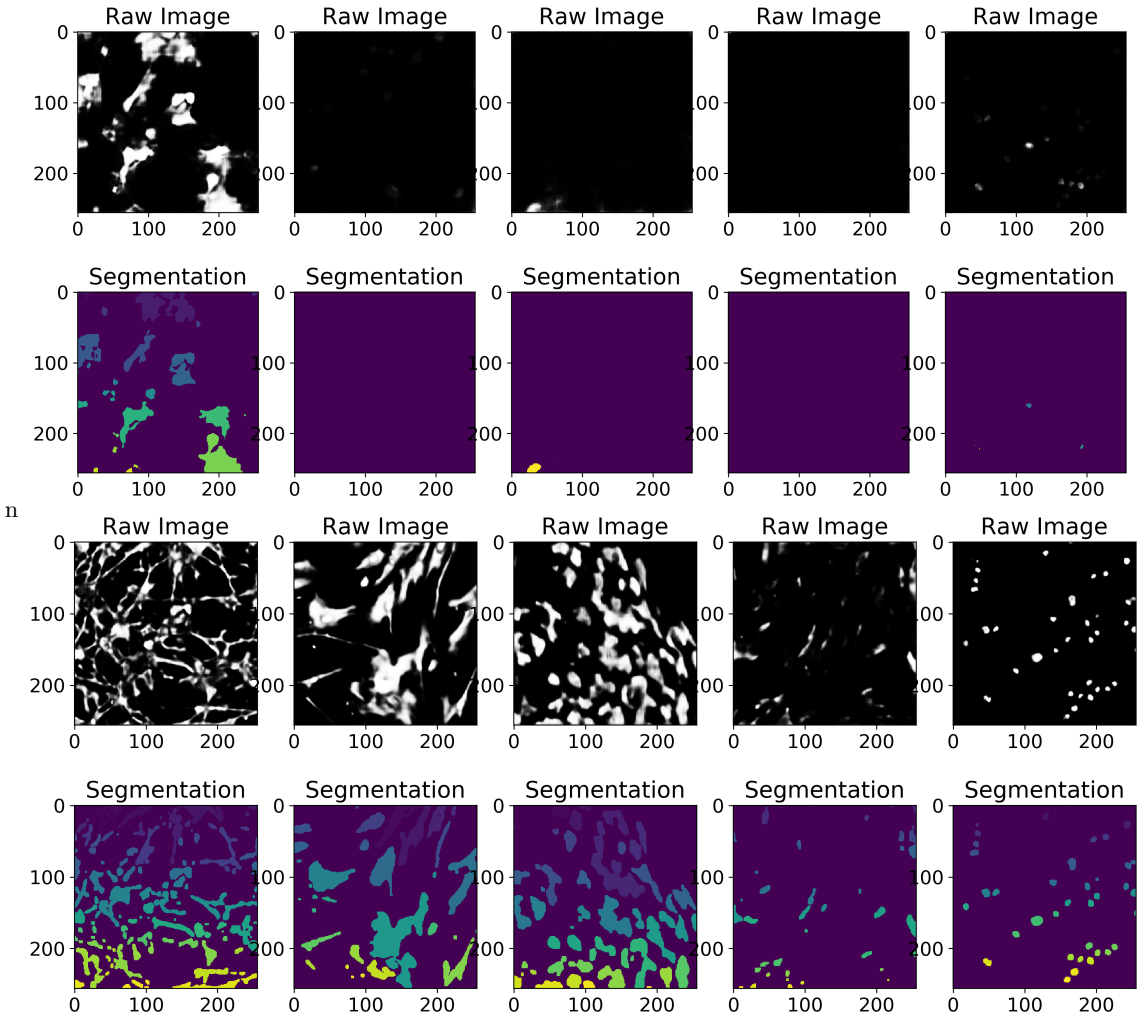

Fig. 6. Segmentation results on stage 2 images: top (with input normalization), bottom (without input normalization).