Gland Segmentation for Histology Images

Koray Yenal

Abstract

In this project, I analyze images of colorectal cancer. Our main task is gland segmentation. For this purpose, I build two custom networks: a main network and a long-skip network. I preprocess images before the network and post-process them for obtaining better results. The advantages and limitations of the used methods are discussed. Two methods are briefly compared and contrasted. I also evaluate the results and suggest ways to improve.

Selected Dataset:

The selected dataset was collected by pathologists at the University Hospitals Coventry and Warwickshire, UK in 2015. [1] The images are H&E stained slides, with varying histologic grades. The examined cancer type is colorectal cancer. The inputs are microscopy images and the desired outputs are masks that label each pixel if it is part of a gland or not. The dataset has ground truth annotations by pathologists. There are 165 images in the dataset. The training set size is 85, of which 37 are benign and 48 are malignant. The training set size is 80, of which 37 are benign and 43 are malignant. Each image has a size of 256x256x3. The image resolution is 0.62005$\boldsymbol{\mu}$/pixel. There is one ground truth object per label. The ground truth information for each image is stored in BMP format.

The selected dataset is significant in the sense that prior to this challenge, gland segmentation had mostly been done in healthy or benign samples. The samples in this dataset include samples with high-grade cancer as well. However, there are two limitations of this dataset: First, there are only 165 images in the dataset. This number is very low, considering that datasets such as ImageNet have millions of images. Second, images are obtained from the same scanner, Zeiss MIRAX MIDI Slide Scanner. The fact that all images are obtained from the same scanner means that there is a high risk of overfitting.

Problem Motivation:

The challenge is to segment glands. The main motivation is the accurate segmentation of glands. The gland segmentation task is challenging because gland morphology is usually irregular and heterogeneous. Glands have different shapes, sizes, and morphological features. Thus, accurate

segmentation is critical to retrieve reliable morphological features afterward. morphological features are then used by pathologists to measure to determine the malignancy level of glands.

Model Flow:

I start by shuffling the entire image set at each epoch. I select N images for each batch. I resize each image and recompile them into a batch. Here, I had to properly choose the size of batches, N. I then use data augmentation to produce batches with minor modifications. In prediction, I resize an image. I choose which kind of overlap I want. I get class probabilities for every pixel. As a result, I obtain a prediction map for the whole image. Then, I compute evaluation metrics on the prediction maps.

Methodology:

I used Google Colab (Python) to obtain the results. I have three main parts to the project. First is the data generator. The data generator reads the images and preprocesses images with methods such as scaling and resizing. It also applies data augmentation to manipulate images to increase the number. I then generate batches of image-target pairs for training, validation, and testing. The second, is the deep learning model. It has inputs of 256x256x3 and outputs of 256x256x2 masks. I train the network to optimize the weights in the network. Third, I have the evaluation stage, where I have target-prediction pairs as input and performance metrics as output. I use three evaluation metrics for this project. First, for detecting individual glands, a metric allied F1-score is defined in the challenge website. [1] However, I prefer to use precision and recall instead of the F1 score. The F1 score is not a good score for segmentation because it is sensitive to imbalanced data and does not really tell us how the algorithm does. Precision and recall in this application are more informative. Second, for segmentation, I use the Matthews correlation coefficient (MCC) metric. MCC's formulation is given below:

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

MCC has two main advantages: First, it is symmetric, meaning that if I switch positives and negatives, I still get the same value. Second, if MCC is close to 1, it means that both classes are predicted well. This makes MCC a very interpretable metric.

Preprocessing:

Most networks require fixed-size inputs. Thus, I resize all images and the labels to a fixed size of input of 256x256x3. The main advantage of resizing is that it is easy to implement and I keep the whole context of the image. The main disadvantages of this method are twofold: First, it might cause image deformation if I do not keep the aspect ratio the same. Second, I lose some image resolution and the

quality may drop. One of the problems I encountered was that the resize method uses some interpolation and rescaling so I might actually lose our labels. To avoid this problem, I specified a few parameters to ensure consistent segmentation. After resizing, I needed to transform them into batches. For that, I randomly selected “n” images to construct a batch with size n. I shuffled the image list at each epoch to prevent ordering.

Then, I implemented data augmentation, to improve the generalization capabilities of the algorithm and solve the insufficient training samples issue. I implemented vertical and horizontal mirroring, Gaussian noise, and gamma correction to compensate for the very low number of images. Essentially, I applied small transformations in the images. For example, I obtained vertical and horizontal mirror images by just reversing the axes of the images. Then, I added some Gaussian noise using NumPy’s random module. The standard deviation of the normal controls the amount of noise. Thus, the standard deviation is one of the parameters of data augmentation. Then, I add some random gamma correction, with random gamma values between 0.5 and 1.5. This gamma randomly brightens or darkens the image.

Network Architecture:

I compare two architectures: the main architecture and a long-skip architecture. For the main model, I use a common architecture with an encoder and a decoder part. In the encoder part, I used convolutional and pooling layers to extract feature maps. In the decoder part, I added convolution and upscaling layers to increase the size to get a mask the same as the image. Here, one choice is how many convolutional layers or how much pooling to make. The decoder part needed enough resolution coming from the encoder part. I also had a tradeoff with the number of features and the number of convolutional layers. The more convolutions, the more overfitting would be. Thus, I needed to balance that. For the long-skip model, in addition to the above-mentioned features, shortcuts are made from the encoder part to the decoder part. The main advantage of using long-skip connections is that it allows us to introduce high-resolution image information into the later stages of the network. This, in turn, improves image segmentation quality.

Our constructed network has a simple structure: two max-pooling layers with two convolutions in between and two upsampling layers in the decoder part. There are 128 features at the end of the encoder. For the activation function, I chose ReLU, a common choice. I used the ‘same’ padding for the borders of the image. For the optimization algorithm, I picked Adam optimizer. I included an early-stopping condition with a certain number of patience. I use patience of 15-20 epochs in our model. If the model does not learn for the time determined by the patience parameter, then I interrupted the model. I used

'sparse categorical cross entropy' as a loss function. With this network, I fit the parameters and get the validation and training performances. I then get the prediction by using the trained network on the test set.

Post-Processing:

For post-processing, I first filtered objects with an area greater than 400. This area is a hyperparameter I had to choose for our post-processing model. After getting labeled regions from segmentation probabilities, I binarized the probabilities. I labeled connected regions. I removed the output noise. I closed any holes within the objects using filled_image of regionprops.

Interpretation of Results:

Overall, I obtained around 75% accuracy in predicting glands (see Appendix 1). I observe that in general, the models perform well on the training set but performance drops on the test and validation sets. This observation is trivial. Second, I observe that the test set performance is as good as the validation set for both models. Third, the long-skip model seems to perform better than the main model. In this segmentation problem, I obtained relatively low MCC results for both the main model and the long-skip model for training and validation (see Appendix 6,7). I obtained around 60-65% recall, which is fairly good. However, our precision was lower than expected. For this reason, I did not obtain the test results for these metrics, but only interpreted what could be the reasons for such low values. One interpretation of such low segmentation results is that the network architecture had difficulty separating adjacent glands. These adjacent glands are usually much harder for algorithms to detect and learn. For mitigating this, as the next step, I could work more on the post-processing. Another solution could be to add a third class, which would be the borders between the glands. This way, the algorithm could learn to learn borders much better by penalizing the network during learning. In addition, low accuracy might be caused by irregularly shaped glands. Another interpretation could be that the pre-processing methods were inadequate and did not diversify images enough. As the next step, I could apply different pre-processing techniques such as rotation or scaling.

In addition, I realize that the modifications I applied in pre-processing and post-processing stages have a significant impact on the results, as there are only 165 images in the dataset. This shows the importance of optimizing the entire pipeline, not just the deep neural network architecture.

References:

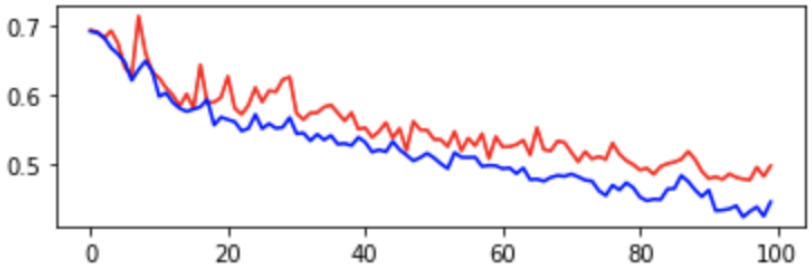
[1] GlaS@MICCAI'2015: Gland Segmentation Challenge Contest, TIA Centre, Dept of Computer Science, University of Warwick https://warwick.ac.uk/fac/cross_fac/tia/data/glascontest/

Appendix:

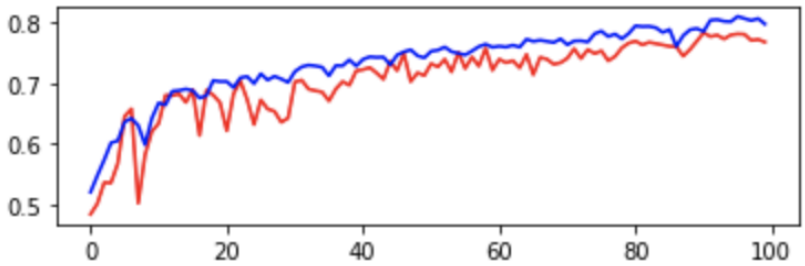
1. *Table of Results*

Accuracy / Model	Training Set	Validation Set	Test Set
Main Model	**76.22**	**72.53**	**73.39**
Long-Skip Model	**80.74**	**77.05**	**77.19**

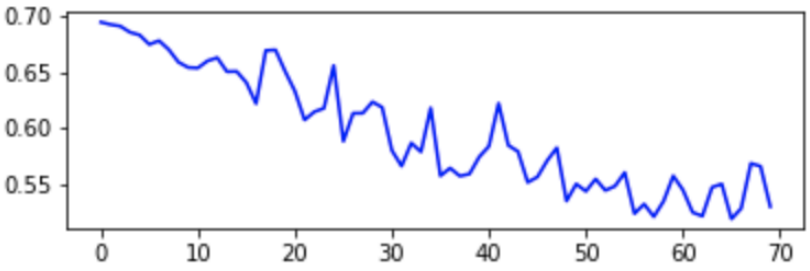
2. *Main model Training-Validation Results (Blue: Training, Red: Validation)*
   a. Entropy Loss



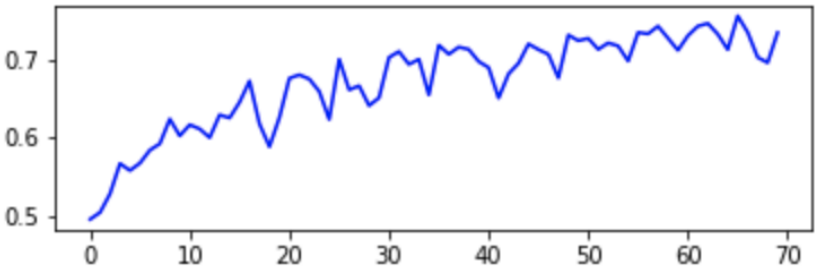
   b. Accuracy



3. *Main model Test Results*
   a. Entropy Loss

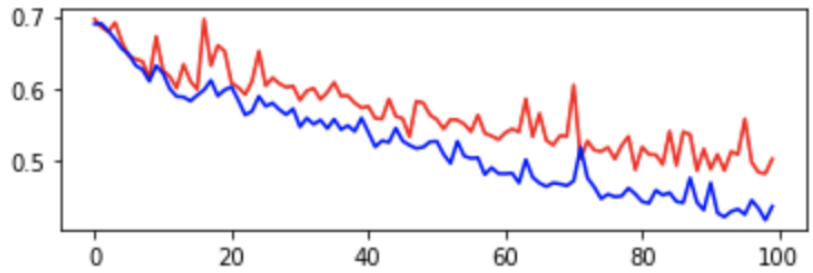


b. Accuracy

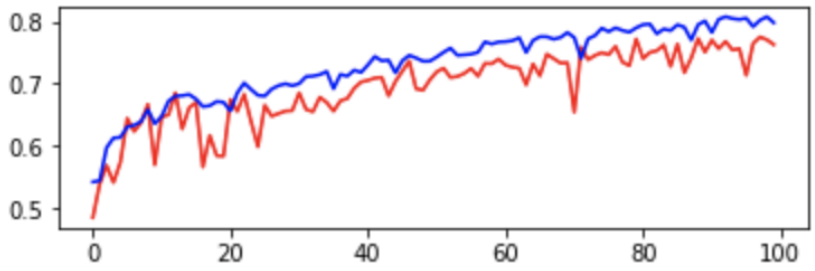


4. Long-Skip Model Training-Validation Results (Blue: Training, Red: Validation)

a. Entropy Loss

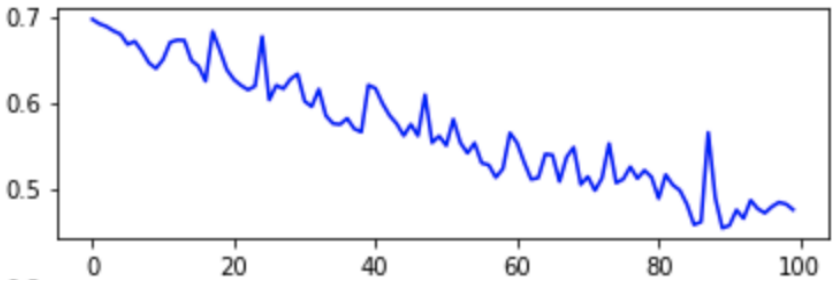


b. Accuracy

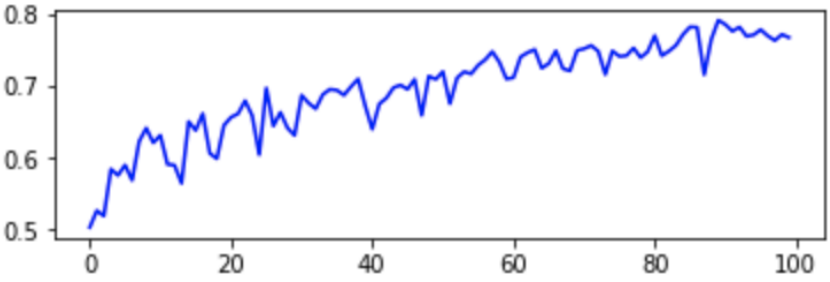


5. Long-Skip Model Test Results

a. Entropy Loss



b. Accuracy



6. *Main Model Precision-Recall-MCC Results*

```
Training performance:
Precision       Recall  MCC
[0.38728062 0.75793848 0.57567813]
[0.4        0.83333333 0.5906403 ]
 ----
Validation perfomance:
Precision       Recall  MCC
[0.33506113 0.73463203 0.55396788]
[0.29051383 0.75649351 0.57825394]
```

7. *Long-Skip Model Precision-Recall-MCC Results*

```
Training performance:
Precision       Recall  MCC
[0.27002485 0.74080128 0.49007294]
[0.25       0.75       0.49120817]
 ----
Validation perfomance:
Precision       Recall  MCC
[0.22105403 0.65145022 0.47021884]
[0.16233766 0.625      0.4429547 ]
```