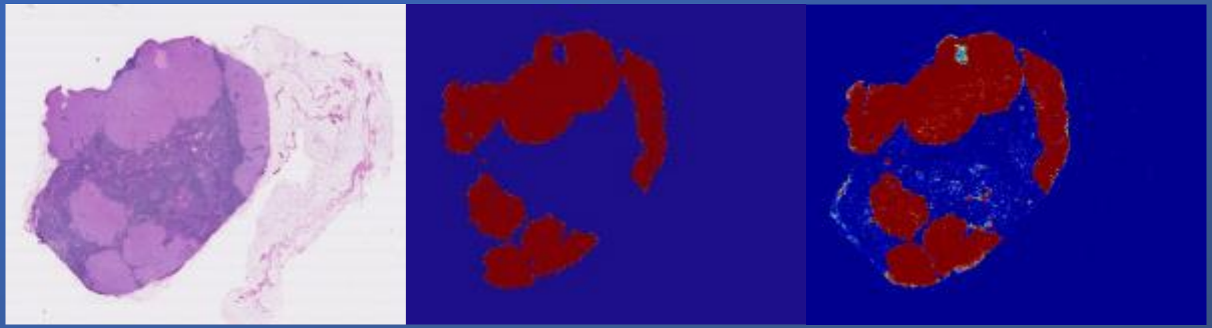
COMSW 4995 - Applied Deeping Learning Project Presentation

Jinxu Xiang (jx2399)

Content

1. Introduction

Goal: Given a collection of training data, develop a model that outputs a heatmap showing regions of a biopsy image likely to contain cancer.



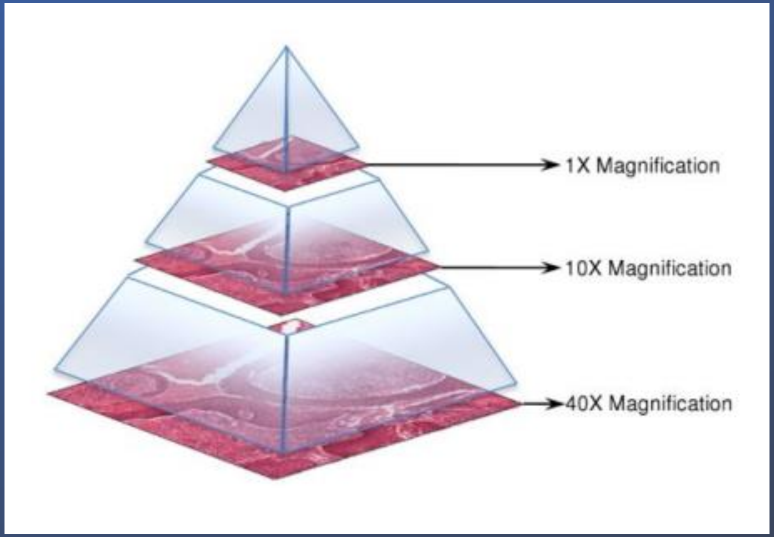
Biopsy image

Ground truth
(from pathologist)

Model predictions

Images are large (> 100k x 100k pixels).

OpenSlide



Each image has multiple resolutions.

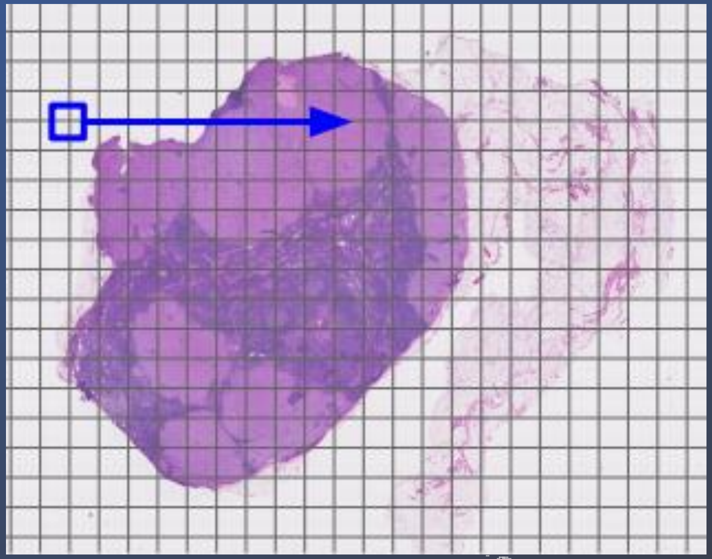
Reference:

1. Lecture 4, Joshua Gordon, COMSW 4995, Fall 2020

2. Cropping Images

There are several methods on cropping images, such as regular grid cropping, sliding window cropping and random cropping. I slid a window across the image and the mask to generate 256*256 images with a 192 pixels sliding distance on both height and width. In addition, I also attached a random movement value to each cropped image. Since the sliding distance is less than the image height, the cropped image area can cover the entire image.

In the image, more than half of the spaces do not contain tissue. To reduce computation, I removed background patches (gray value > 0.8 and verified visually that lymph node tissue was not discarded)



Reference:

1. Lecture 4, Joshua Gordon, COMSW 4995, Fall 2020

3. Building Data Pipeline

The existing training dataset is images under different zoom levels. What we need to send to the model is a batch of the images under the multi-level and the corresponding masks. Here I used level2, level4 and level6 images as input data. All of them contains information about the same area under different resolutions.

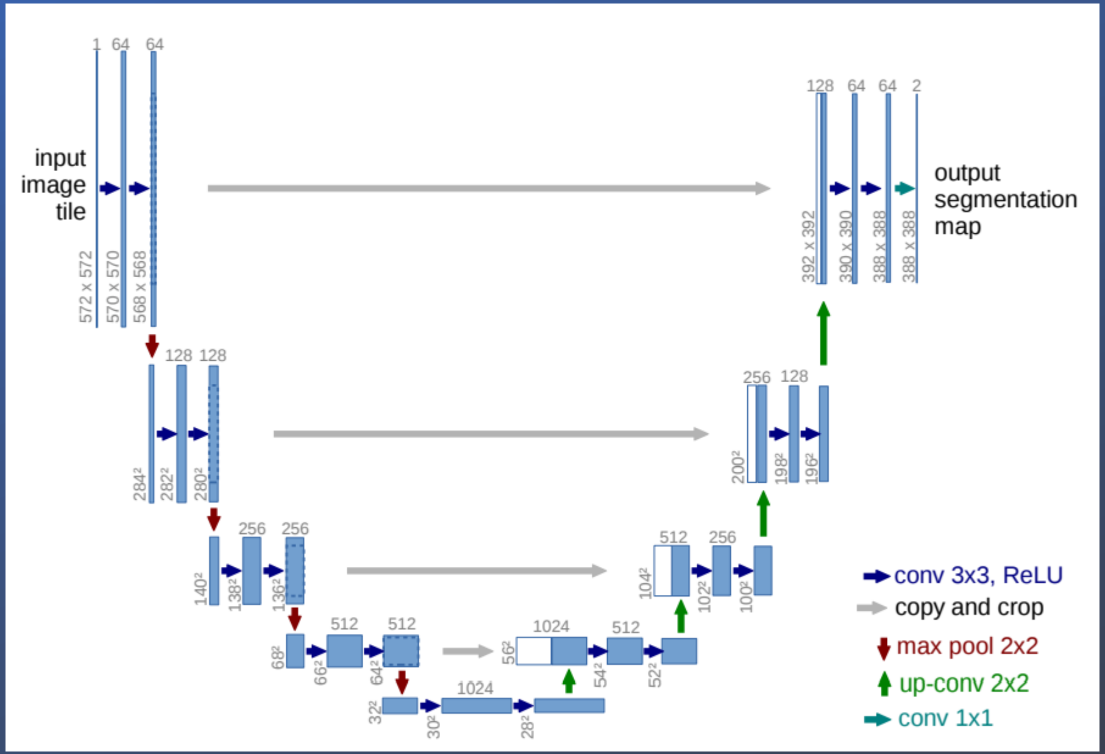
Input dimension and dtype:
batch_slide1 : 32*256*256*3, tf.float64
batch_slide2 : 32*64*64*3, tf.float64
batch_slide3 : 32*16*16*3, tf.float64
batch_mask1 : 32*256*256*1, tf.int32
batch_mask2 : 32*64*64*1, tf.int32
batch_mask3 : 32*16*16*1, tf.int32

```
def DataGenerator(batch_size , category = ‘train’/’valid’/’test’):
  Calculate total number of cropping area and the cropping position in ‘category’ data.
  while True:
    Select 1 data augmentation method
    for i in range(batch_size):
      r = random position of cropping area
      slide1,2,3 = read slide in level 2,4,6 at position r
      mask1,2,3 = read mask in level 2,4,6 at position r
      for j in range(n):
        batch_slide1,2,3 [i*n+j] = nth data augmentation on slide1,2,3
        batch_mask1,2,3 [i*n+j] = nth data augmentation on mask1,2,3
    yield (batch_slide1, batch_slide2, batch_slide3), (batch_mask1, batch_mask2, batch_mask3)
```

4. Modifying Model

UNet Architecture

The architecture looks like a ‘U’ which justifies its name. This architecture consists of three sections: The contraction, The bottleneck, and the expansion section. The contraction section is made of many contraction blocks. The number of kernels or feature maps after each block doubles so that architecture can learn the complex structures effectively. The bottommost layer mediates between the contraction layer and the expansion layer. The heart of this architecture lies in the expansion section. Similar to contraction layer, it also consists of several expansion blocks. It also concatenate the input from contraction layers.

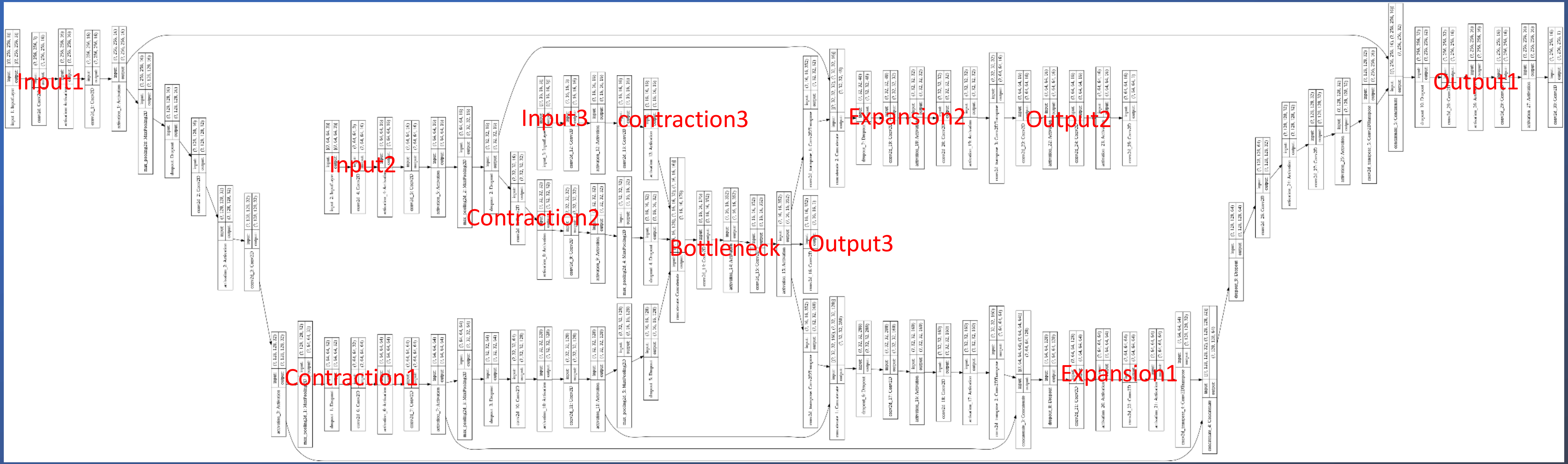


Reference:

1. U-Net: Convolutional Networks for Biomedical Image Segmentation. Olaf Ronneberger, 2015

4. Modifying Model

My model



(256, 256, 3) -> (256, 256, 16) -> (128, 128, 16) -> (128, 128, 32) -> …… -> (32, 32, 128) --
(64, 64, 3) -> (64, 64, 16) -> (32, 32, 16) -> (32, 32, 32) --
(32, 32, 160) -> (16, 16, 160) --
(16, 16, 3) -> (16, 16, 16) --
(16, 16, 176) -> (16, 16, 352) --
(32, 32, 160) -> …… ->
(16, 16, 1)
-> …… -> (256, 256, 1)
(64, 64, 1)

Reference:

1. U-Net: Convolutional Networks for Biomedical Image Segmentation. Olaf Ronneberger, 2015

5. Creating the Loss Functions

Because the tumor area occupies a small proportion in the whole image, we cannot directly use Binary Crossentropy Loss.

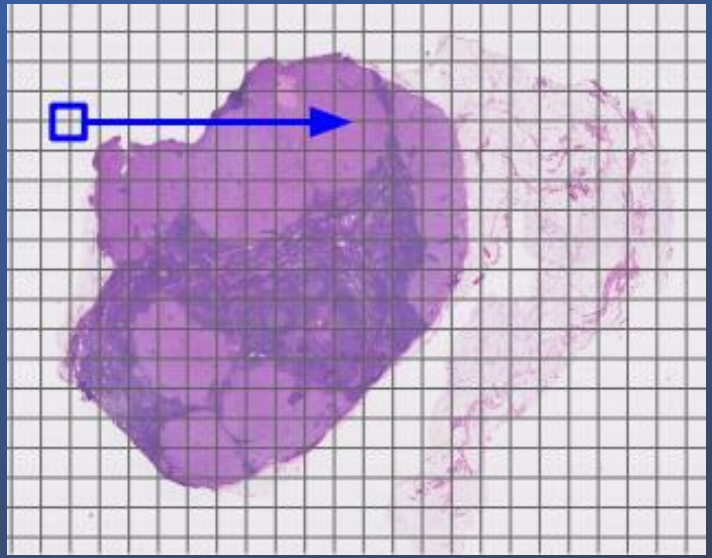
We need to adjust the coefficient of Loss according to the area of the tumor area. For example, if 1 represents the tumor area, 0 represents the normal area, and the tumor area only occupies 10% of the area, then the loss of predicting 0 into 1 needs to be smaller and 1 into 0 should be larger. I choose this coefficient according to the area proportion.

```
def focal_loss(gamma=2., alpha=.25, coeff=1):

  def focal_loss_fixed(y_true, y_pred):
    pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
    pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
    return (-K.sum(alpha*K.pow(1.-pt_1, gamma)*K.log(K.mean(pt_1))) -
          K.sum((1-alpha)*K.pow(pt_0, gamma)*K.log(K.mean(1.-pt_0))))*coeff

  return focal_loss_fixed
```

And because the size of the input images is inconsistent, for example, a 16*16 picture contains all the information of a 256*256 picture, but the pixel size is only 1/256 of the original, so a large coefficient is needed for keeping the total loss on each image roughly the same.

6. Predicting Tumor Area

We also need to crop the image into small pieces for prediction and exclude areas without cells. Although I did not use the low zoom level to observe a wider area to assist in prediction, the same effect can be achieved by displaying and predicting an area in different positions of the cropped 256*256 image. I used a cropping method with a sliding distance of 128 to predict the image.

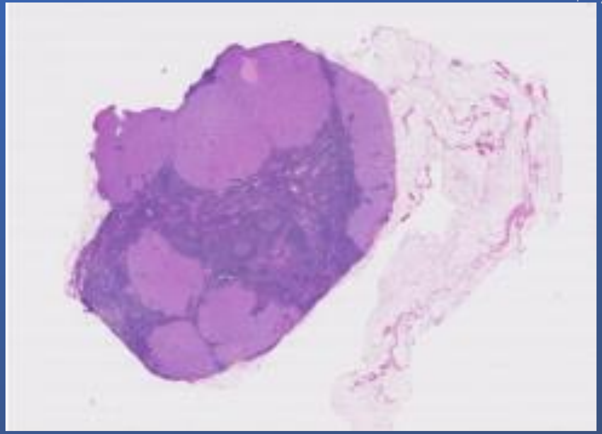


6. Predicting Tumor Area

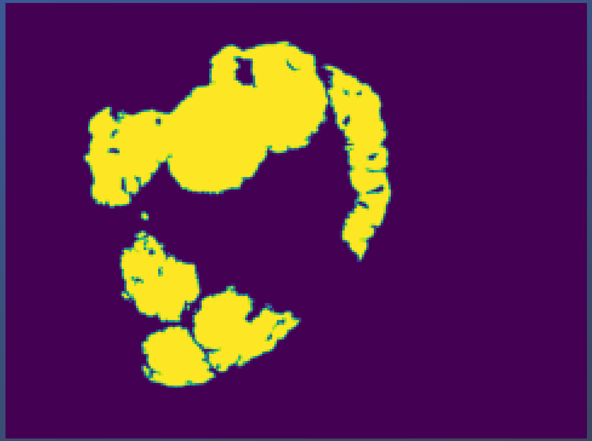
Image 110

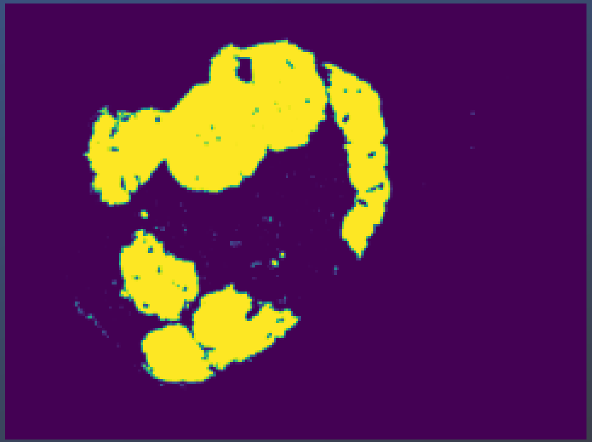
Mask Area: 29.459%
Accuracy: 98.220%

TP: 95.824%
TN: 99.221%
FP: 0.779%
FN: 4.176%



Image


Mask

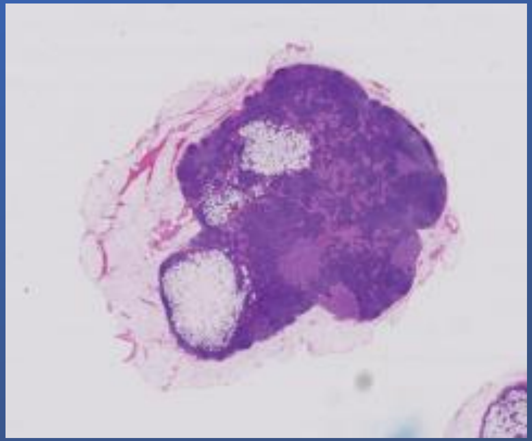

Prediction

6. Predicting Tumor Area

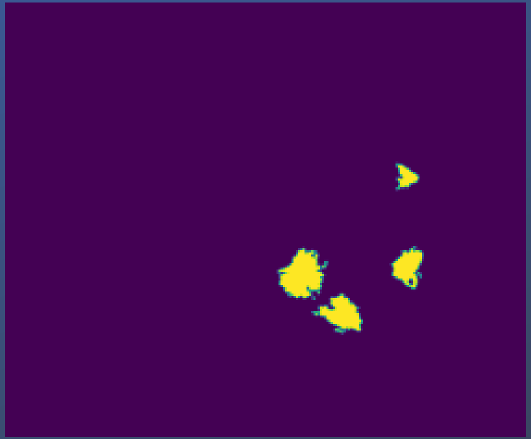
Image 91

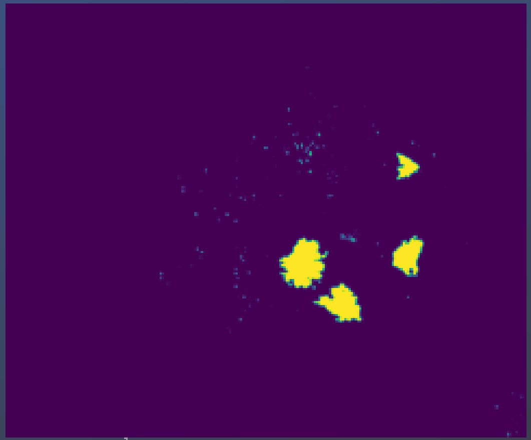
Mask Area: 3.207%
Accuracy: 99.499%

TP: 87.608%
TN:99.892%
FP:0.107%
FN:12.392%

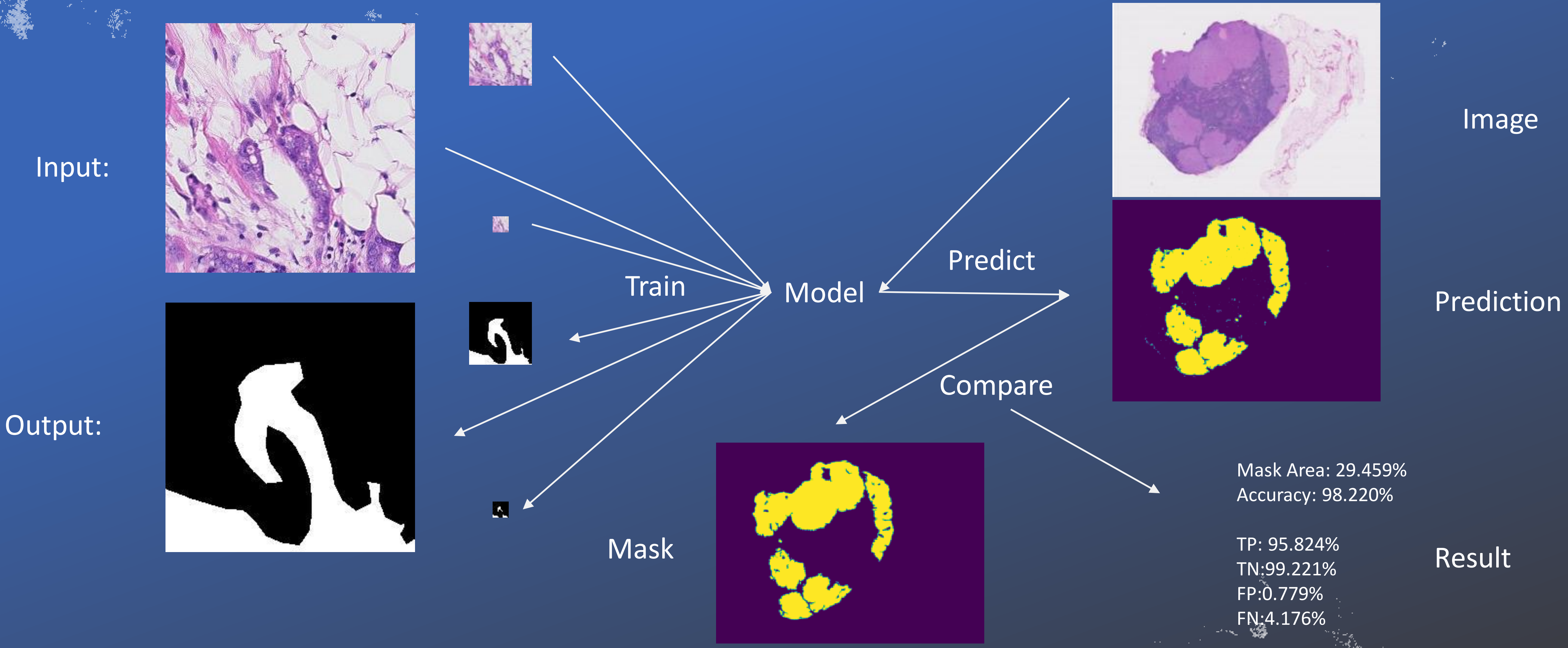


Image


Mask


Prediction

7. Conclusions



Mask Area: 29.459%
Accuracy: 98.220%

TP: 95.824%
TN:99.221%
FP:0.779%
FN:4.176%

Thank you