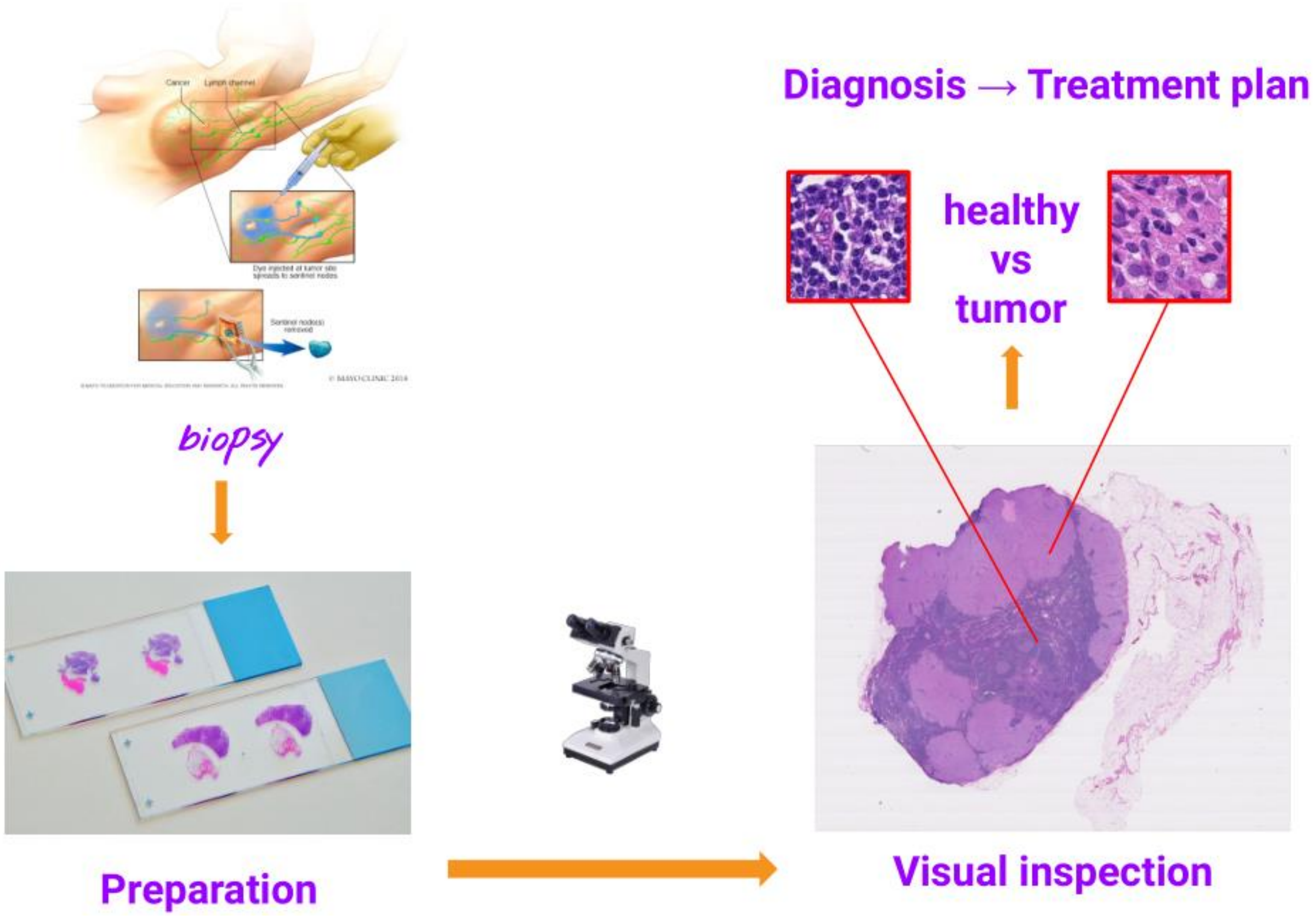
ADL Project Presentation

Detecting cancerous cells in gigapixel images

Zhizhen Cai zc2492

Context (Goal)

For cancer diagnosing!



biopsy

Preparation

Visual inspection

healthy vs tumor

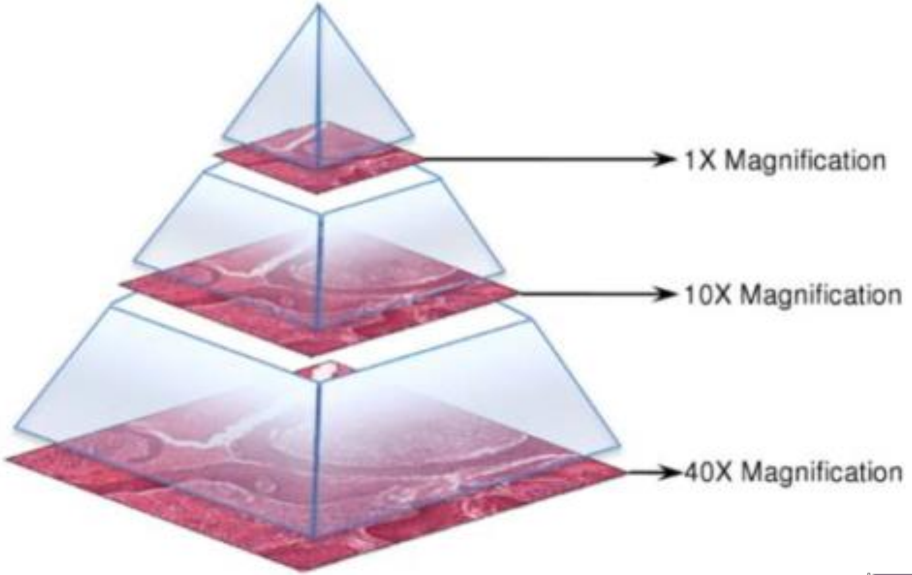
Diagnosis → Treatment plan

Problem: Pathologists make mistakes when diagnosing biopsied tissue

Solve: Develop a tool to assist physicians and reduce misdiagnosis rate

Context (Dataset)

Dataset is from **CAMELYON16**: https://camelyon16.grand-challenge.org/Data/
(given by Professor Joshua)

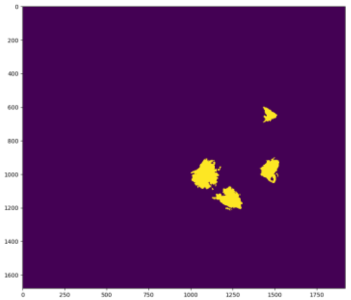


Different Pathology Images

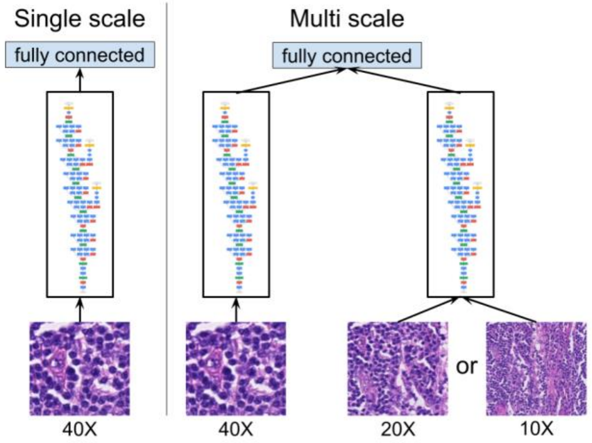
Different resolution levels(0-7)

Has both cells images and tumor mask

Tumor mask: channel 0, value of tumor pixel is 1, other is 0



Methodology



single or multi → Input(s)

① different Pathology Images with one/different resolution levels → extract → patches → Input(s) → CNN

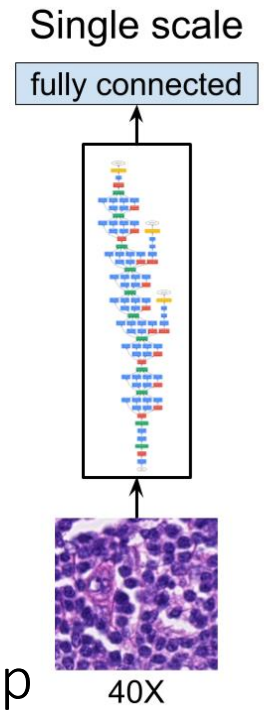
② unseen Pathology Images patches (consistent with training) → CNN → predict tumor location → visualize → heatmap

end-to-end prototype of detecting cancer !!!

Single Input

① different Pathology Images with specific level (level 4) —extract→ patches —single ↓ Input→ CNN

② unseen Pathology Images patches (consistent with training) —CNN→ predict tumor location —visualize→ heatmap



end-to-end prototype of detecting cancer !!!

Extract patches---predefined functions

Patch size: 299 x 299
Center size: 128 X 128

def read_slide: convert a region of slide to RGB array

def find_tissue_pixels: get coordinates of tissue in this region

def apply_mask: return a image where the color of tissue pixel is (1,0,0), other are grey

def get_patches:
Get image patches, image mask patches and tissue patches based on the center (x0, y0) at the specific resolution level

def check_center:
Check whether the center of a patch has tumor

Extract patches---collect patches

Patch size: 299 x 299
Center size: 128 X 128

def get_dataset:
>>> Input: level, patch_size, patch_center, select_image (list)
>>> output: dataset_patch, dataset_label

Detail, for each image:
1. Get tumor and health pixels coordinates from the single resolution level
2. Pick 500 samples in both tumor and health pixels as (x0, y0)
3. If the tissue in the related patch is more than 50% of this patch, collect this patch, 100
4. Record the related label (tumor or health) of this patch

To Colab

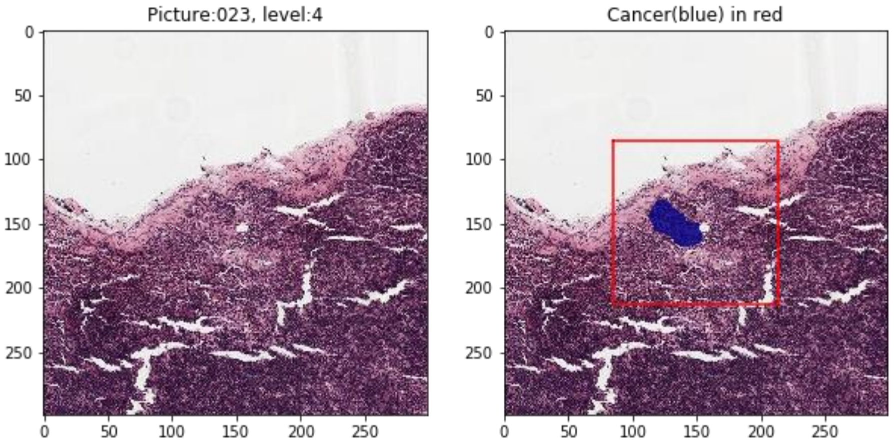
Extract patches---collect patches

Patch size: 299 x 299
Center size: 128 X 128

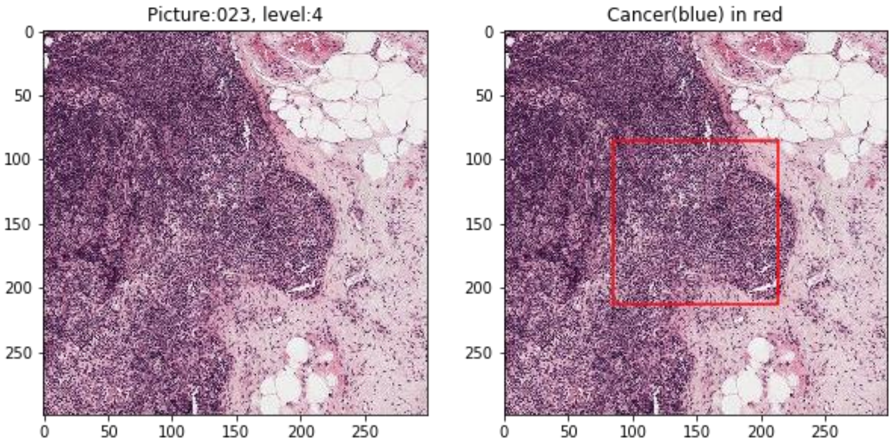
Training images: (10)
'016','023','035','057','059','064','075','081','094','096‘
Test image: (1)
'110'

So I totally have 200*10 = 2000 training samples
200 test samples

tumor



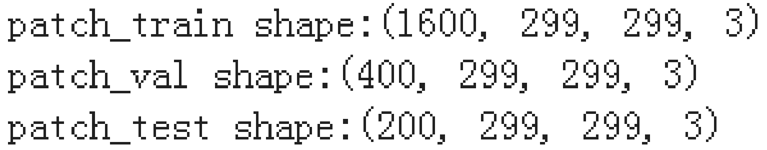
health



Preprocessing

Patch size: 299 x 299
Center size: 128 X 128

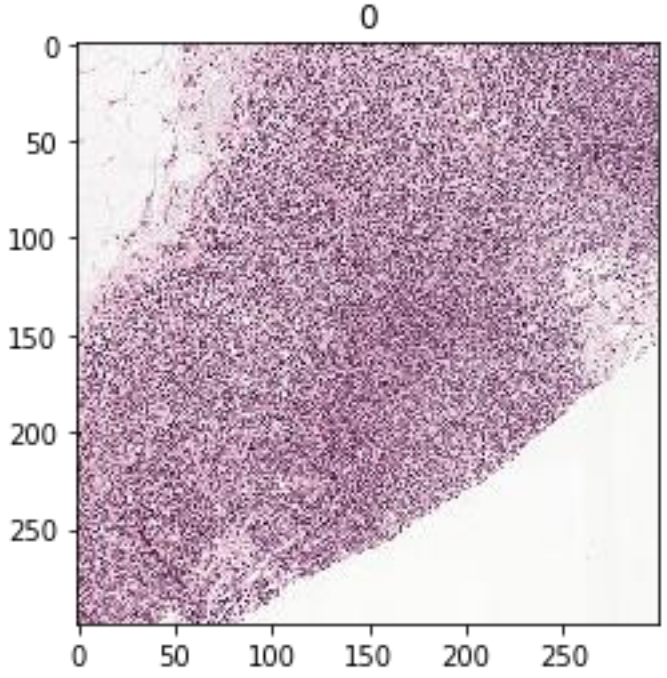
Split training data
training : validation = 8 : 2



```
patch_train shape:(1600, 299, 299, 3)
patch_val shape:(400, 299, 299, 3)
patch_test shape:(200, 299, 299, 3)
```

rescale image to [0,1]

set batch = 32

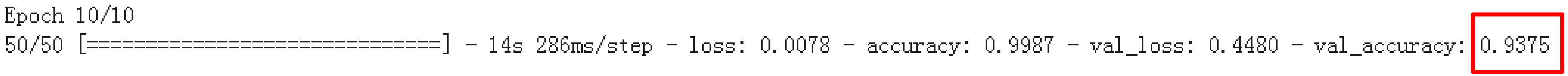


Model

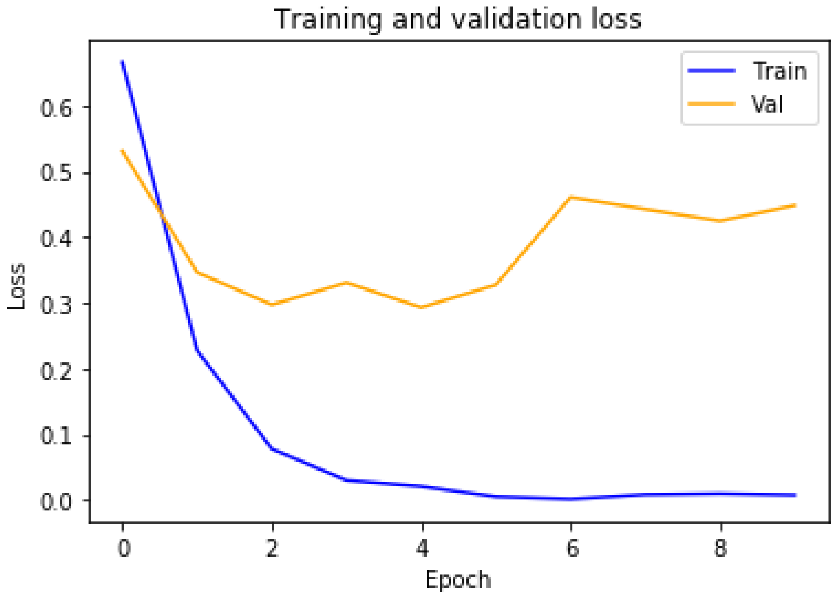
To Colab

Patch size: 299 x 299
Center size: 128 X 128

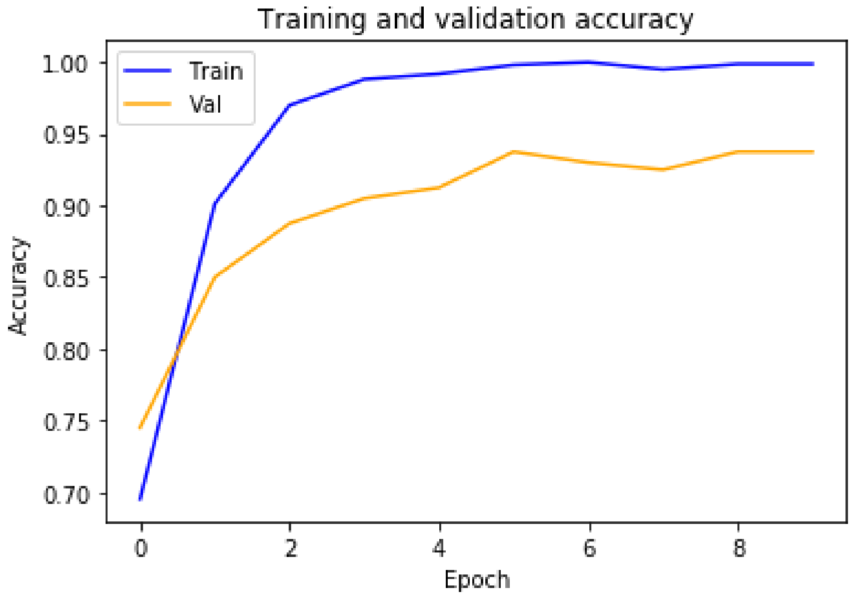
Use CNN model defined by myself:



```
Epoch 10/10
50/50 [==============================] - 14s 286ms/step - loss: 0.0078 - accuracy: 0.9987 - val_loss: 0.4480 - val_accuracy: 0.9375
```



Training and validation loss

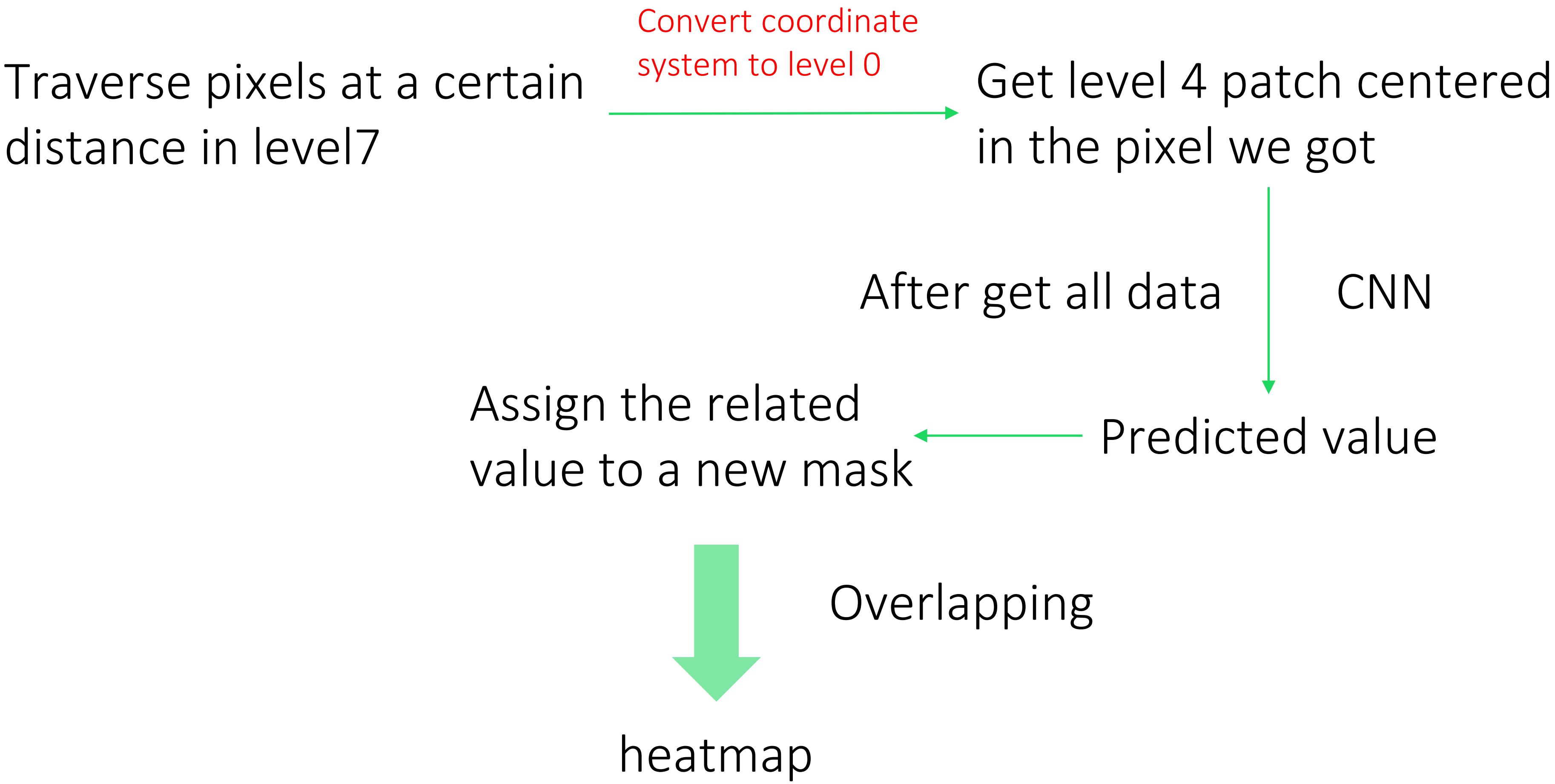


Training and validation accuracy

Evaluate the model: Test loss: 0.280, accuracy: 0.835

end-to-end heatmap

To Colab



Traverse pixels at a certain distance in level7

Convert coordinate system to level 0

Get level 4 patch centered in the pixel we got

CNN

After get all data

Predicted value

Assign the related value to a new mask

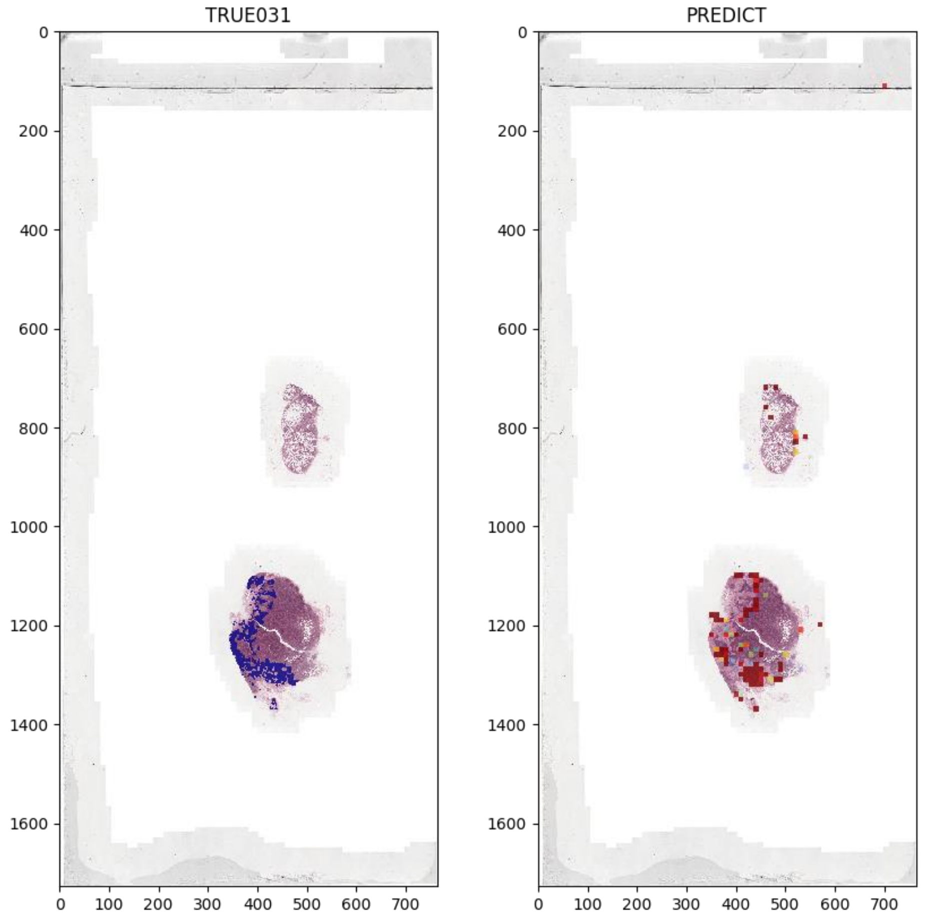
Overlapping

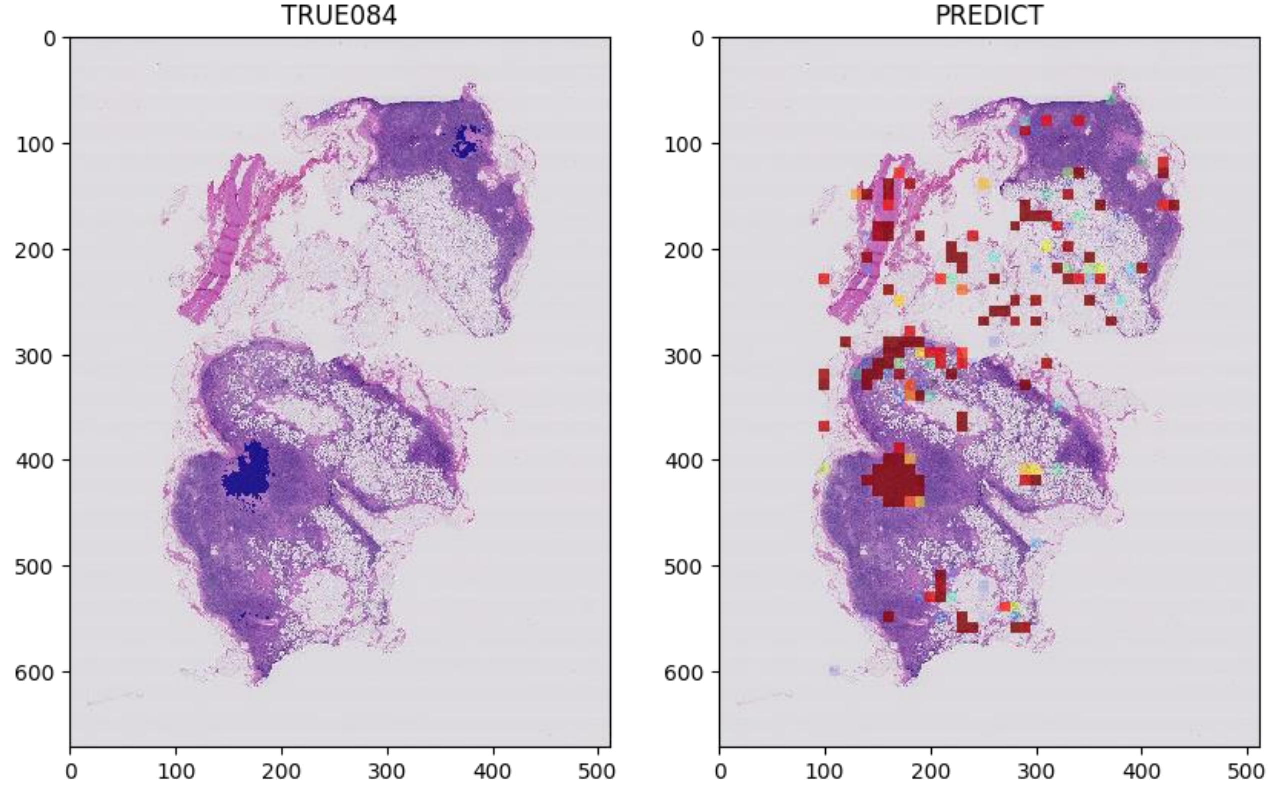
heatmap

end-to-end prototype of detecting cancer !!!

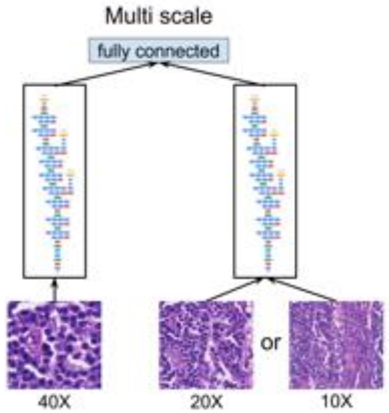
end-to-end heatmap

Heapmap with single input (unseen data)





Multi Input



Multi
↓

① different Pathology Images with 3 levels (level 4 5 6) → extract → Patches same center → Input(s) → CNN

② unseen Pathology Images patches (consistent with training) → CNN → predict tumor location → visualize → heatmap

↓

end-to-end prototype of detecting cancer !!!

Extract patches---collect patches

Patch size: 299 x 299
Center size: 128 X 128

def get_dataset:
>>> Input: level, patch_size, patch_center, select_image (list)
>>> output: dataset_patch, dataset_label

Detail, for each image:
1. Get tumor and health pixels coordinates from the single resolution level
2. Pick 500 samples in both tumor and health pixels as (x0, y0)
3. If the tissue in the related patch is more than 50% of this patch, collect this patch
3.5 add the patch in level 5 and 6 with same size, centered in same (x0, y0)
4. Record the related label (tumor or health) of this patch

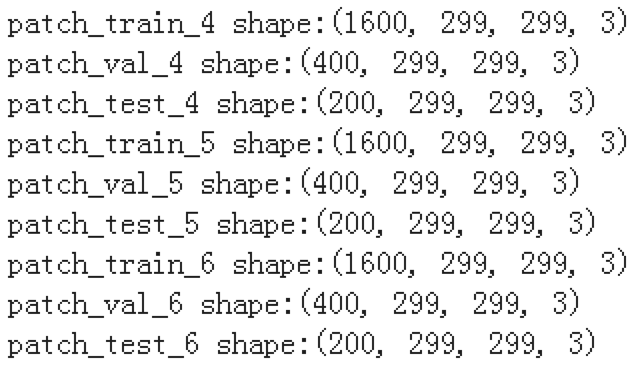
To Colab

Extract patches---collect patches

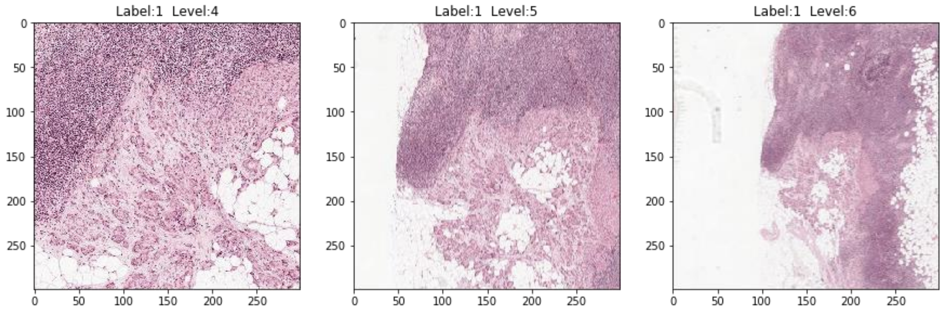
Patch size: 299 x 299
Center size: 128 X 128

So I totally have 200*10*3 = 6000 training samples
600 test samples

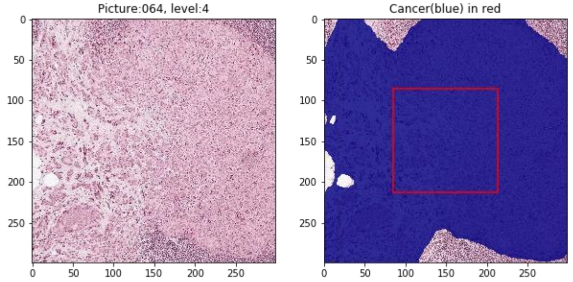
After split:



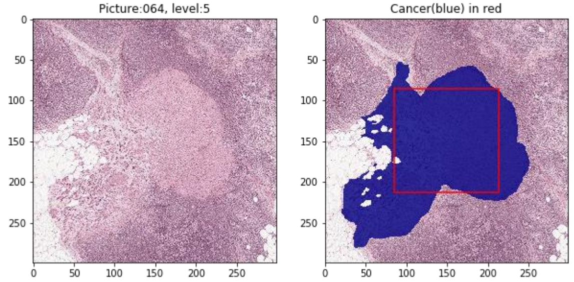
```
patch_train_4 shape:(1600, 299, 299, 3)
patch_val_4 shape:(400, 299, 299, 3)
patch_test_4 shape:(200, 299, 299, 3)
patch_train_5 shape:(1600, 299, 299, 3)
patch_val_5 shape:(400, 299, 299, 3)
patch_test_5 shape:(200, 299, 299, 3)
patch_train_6 shape:(1600, 299, 299, 3)
patch_val_6 shape:(400, 299, 299, 3)
patch_test_6 shape:(200, 299, 299, 3)
```



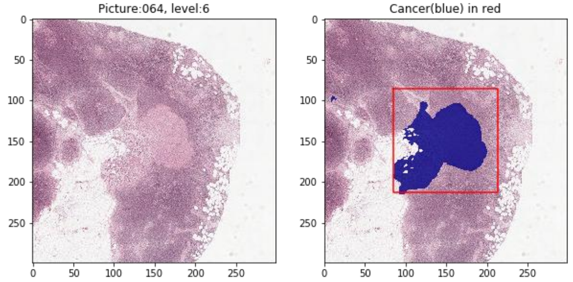
Tumor in level 4



Tumor in level 5



Tumor in level 6

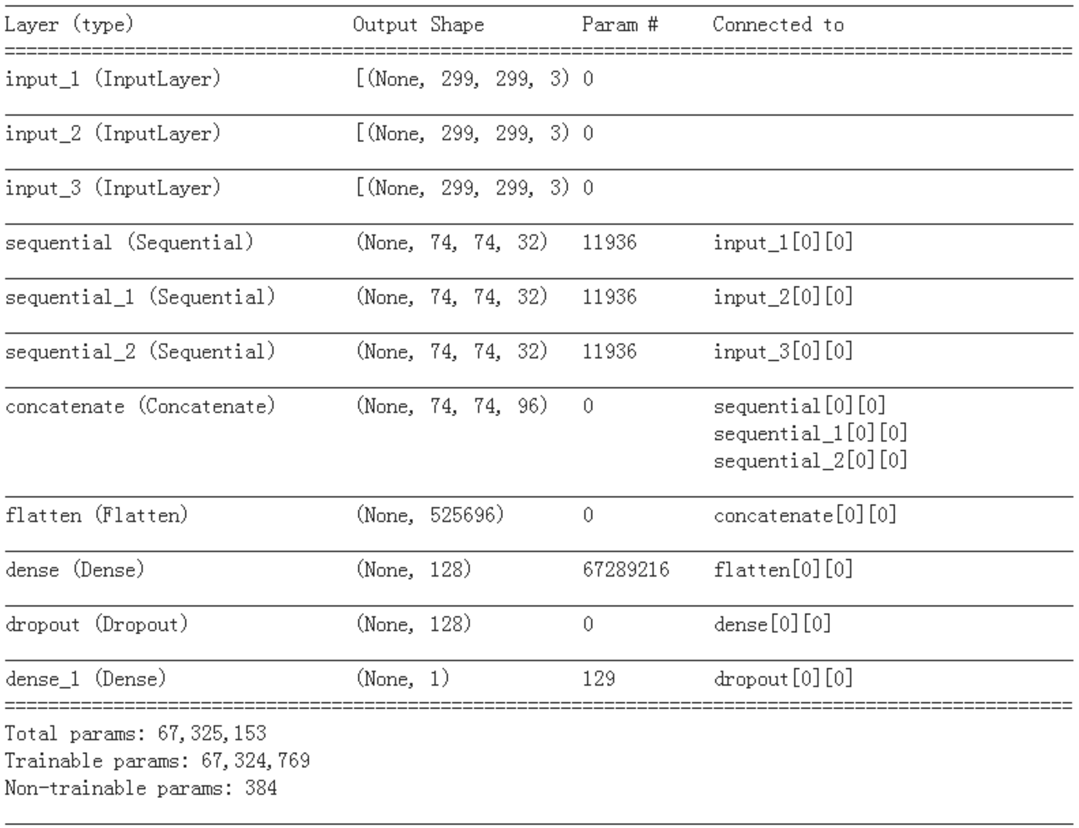


Model

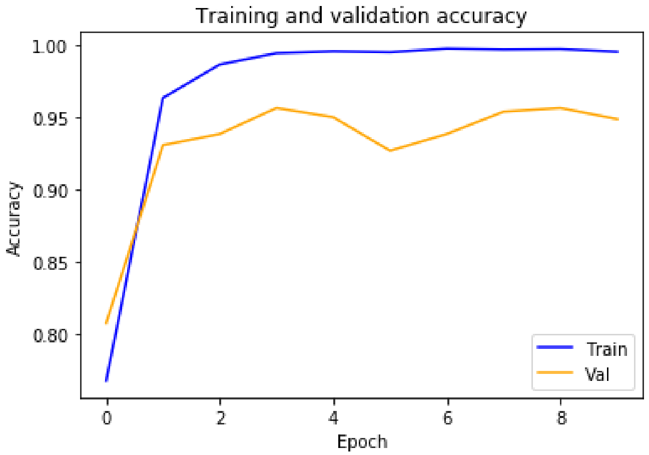
To Colab

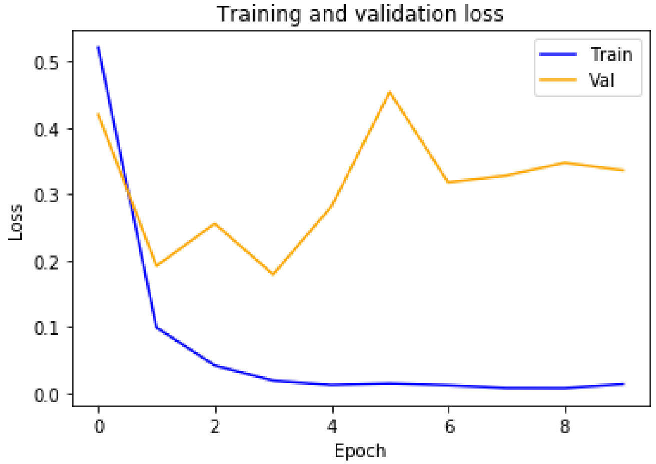
Patch size: 299 x 299
Center size: 128 X 128

Use CNN model defined by myself: have three inputs



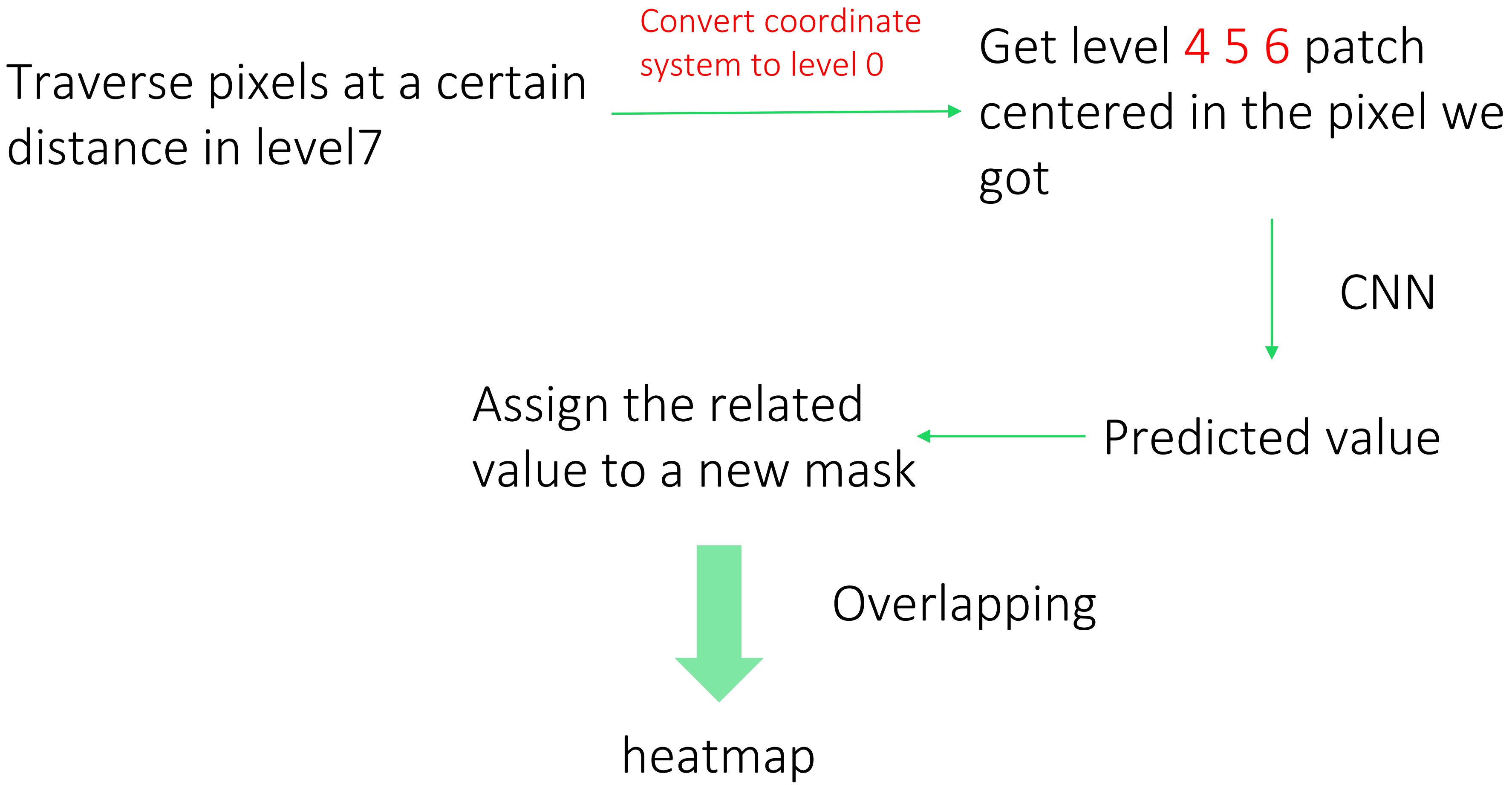
```
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_1 (InputLayer)            [(None, 299, 299, 3) 0
__________________________________________________________________________________________
input_2 (InputLayer)            [(None, 299, 299, 3) 0
__________________________________________________________________________________________
input_3 (InputLayer)            [(None, 299, 299, 3) 0
__________________________________________________________________________________________
sequential (Sequential)         (None, 74, 74, 32)   11936       input_1[0][0]
__________________________________________________________________________________________
sequential_1 (Sequential)       (None, 74, 74, 32)   11936       input_2[0][0]
__________________________________________________________________________________________
sequential_2 (Sequential)       (None, 74, 74, 32)   11936       input_3[0][0]
__________________________________________________________________________________________
concatenate (Concatenate)       (None, 74, 74, 96)   0           sequential[0][0]
                                                                 sequential_1[0][0]
                                                                 sequential_2[0][0]
__________________________________________________________________________________________
flatten (Flatten)               (None, 525696)       0           concatenate[0][0]
__________________________________________________________________________________________
dense (Dense)                   (None, 128)          67289216    flatten[0][0]
__________________________________________________________________________________________
dropout (Dropout)               (None, 128)          0           dense[0][0]
__________________________________________________________________________________________
dense_1 (Dense)                 (None, 1)            129         dropout[0][0]
==========================================================================================
Total params: 67,325,153
Trainable params: 67,324,769
Non-trainable params: 384
__________________________________________________________________________________________
```





Evaluate the model: Test loss: 1.938, accuracy: 0.809

end-to-end heatmap



Traverse pixels at a certain distance in level7

Convert coordinate system to level 0

Get level 4 5 6 patch centered in the pixel we got

CNN

Predicted value

Assign the related value to a new mask

Overlapping

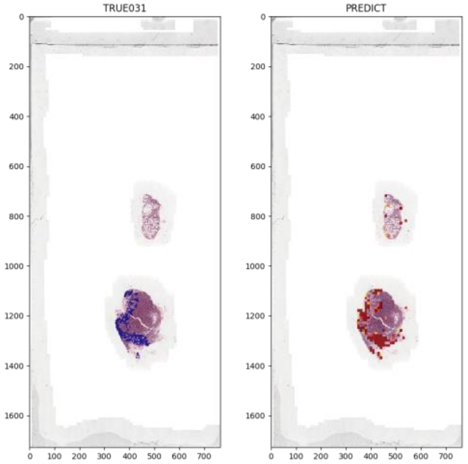
heatmap

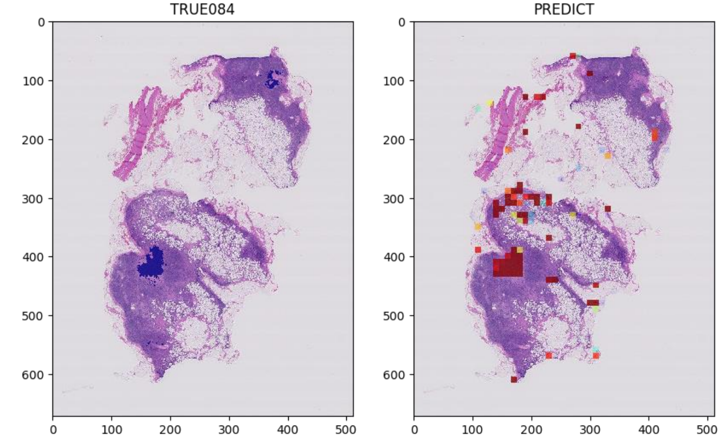
end-to-end prototype of detecting cancer !!!

end-to-end heatmap

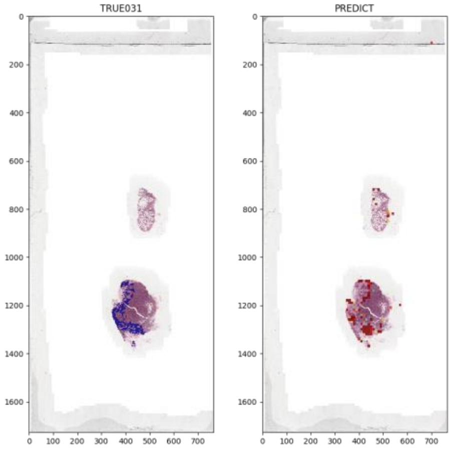
Heapmap with multi inputs (unseen data)

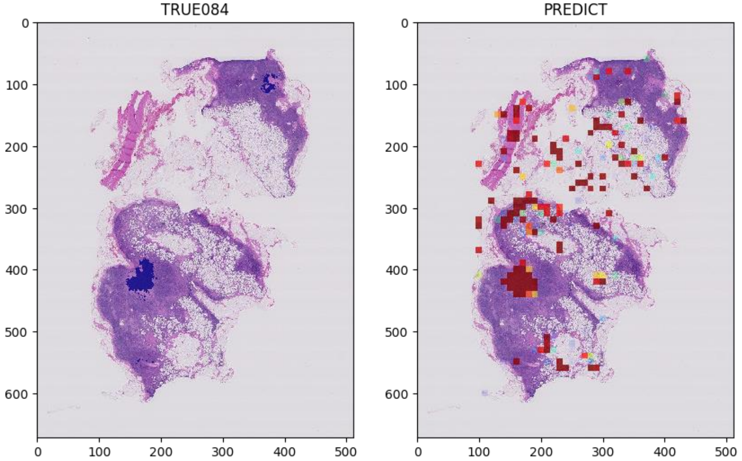
multi





single

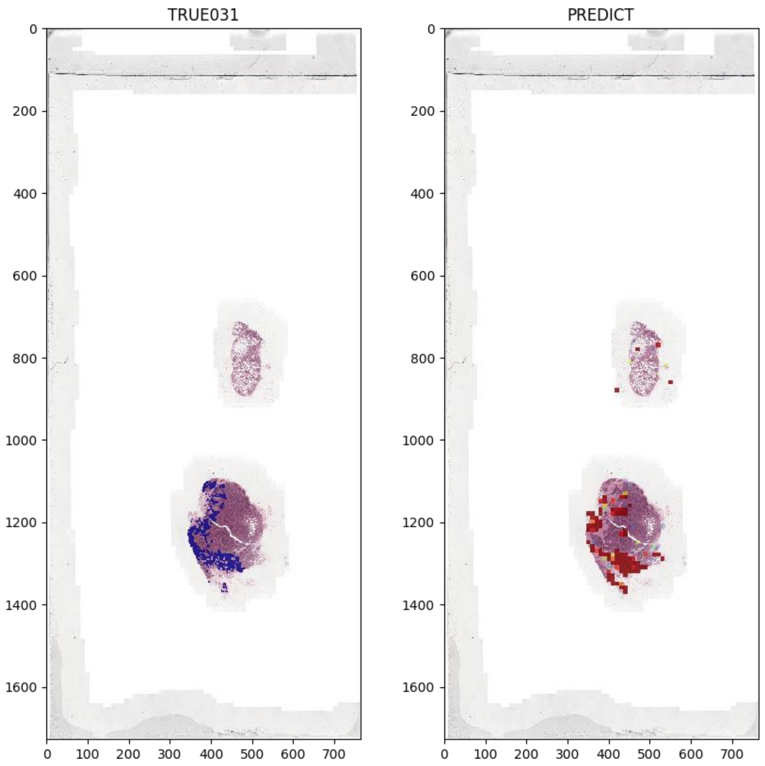


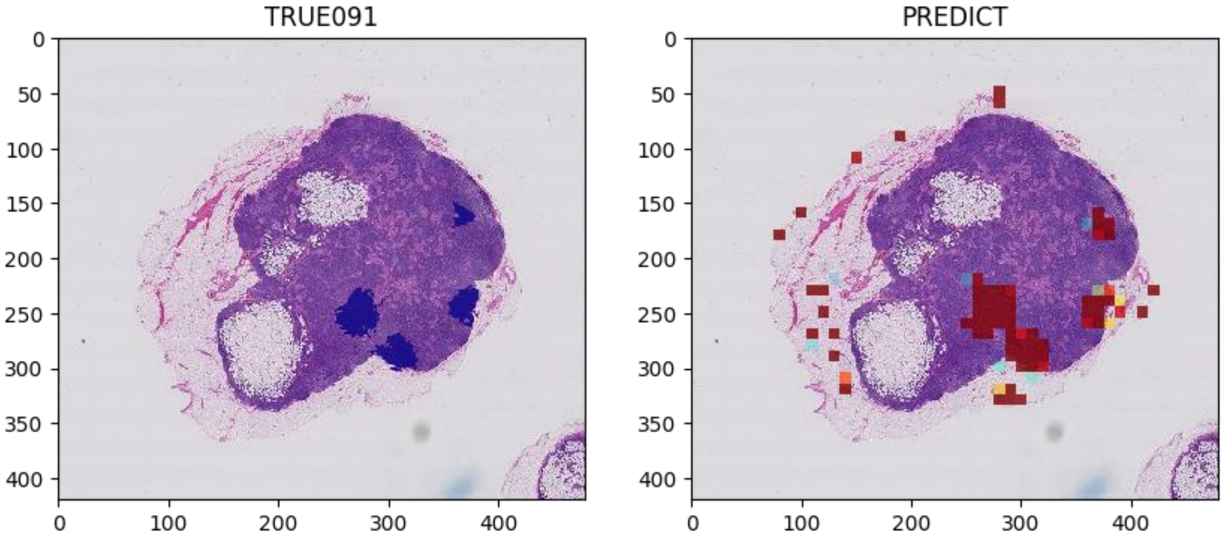


Multi perform better than single

end-to-end heatmap

More heapmap with multi inputs (unseen data)





Improve

Increase the number of training set

Use data augmentation to generate more data since tumor number is small

Thank you