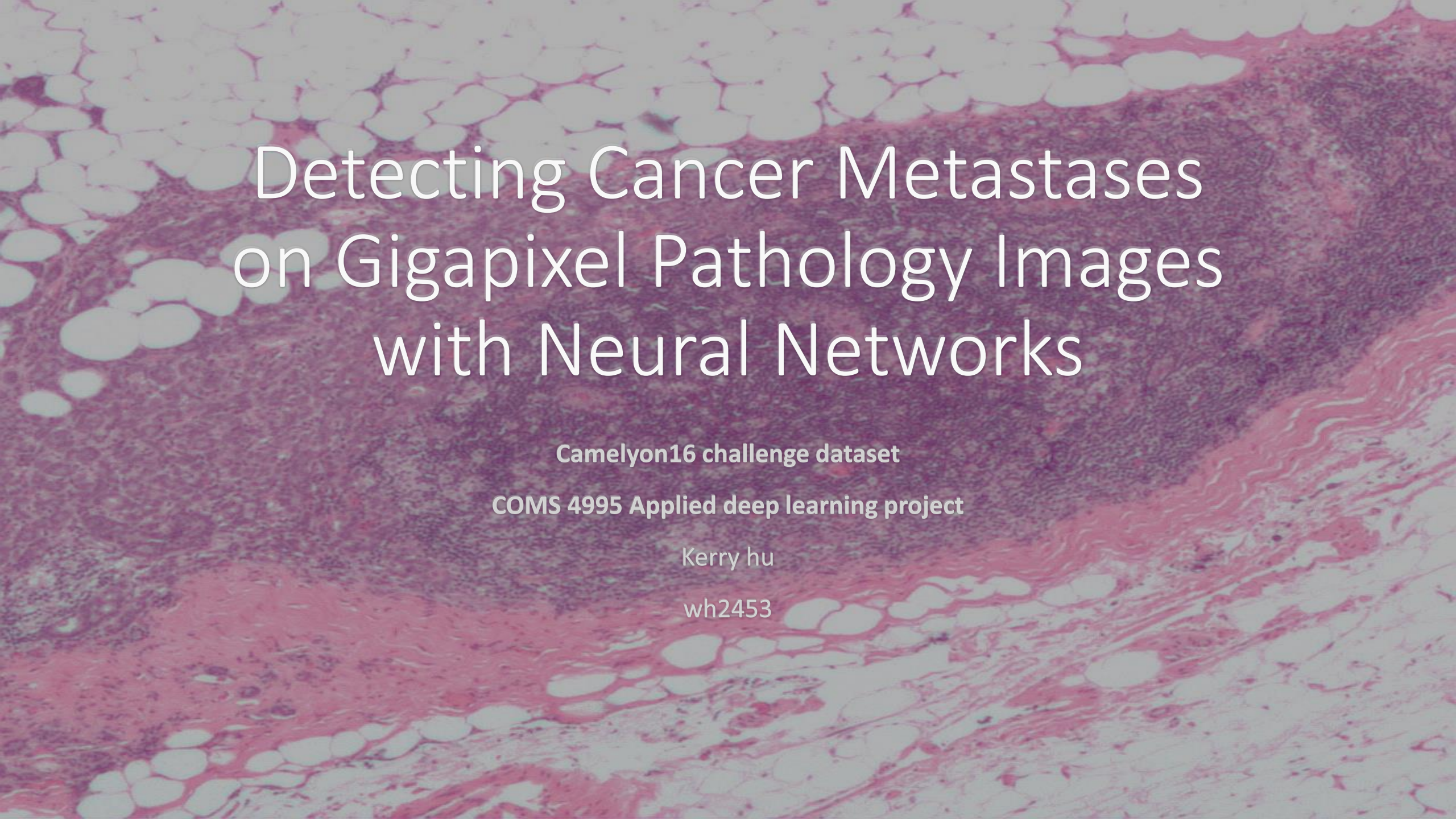


Detecting Cancer Metastases on Gigapixel Pathology Images with Neural Networks

Camelyon16 challenge dataset

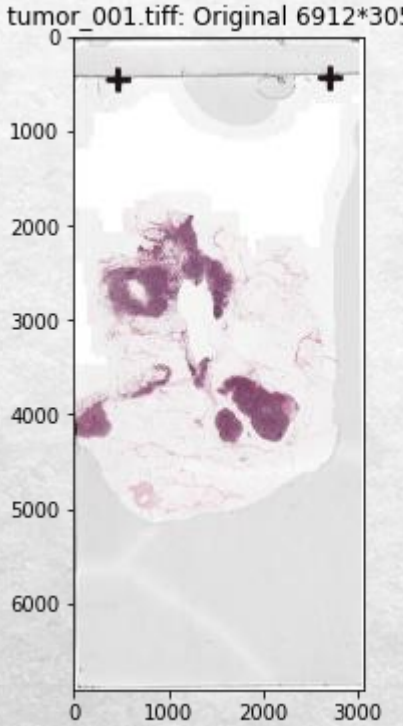
COMS 4995 Applied deep learning project

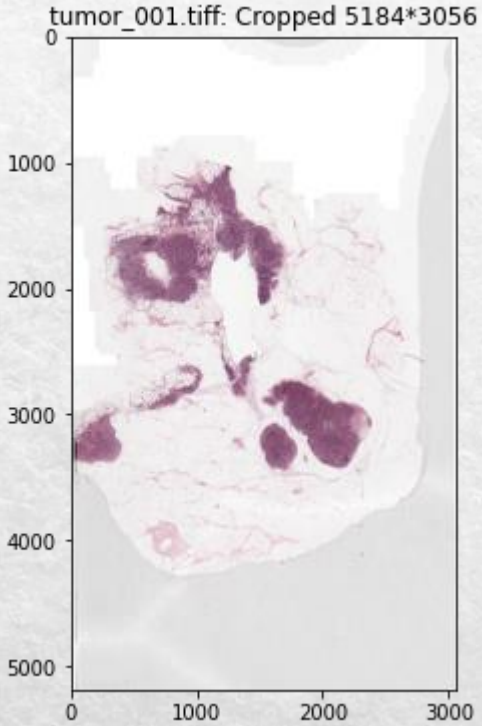
Kerry hu

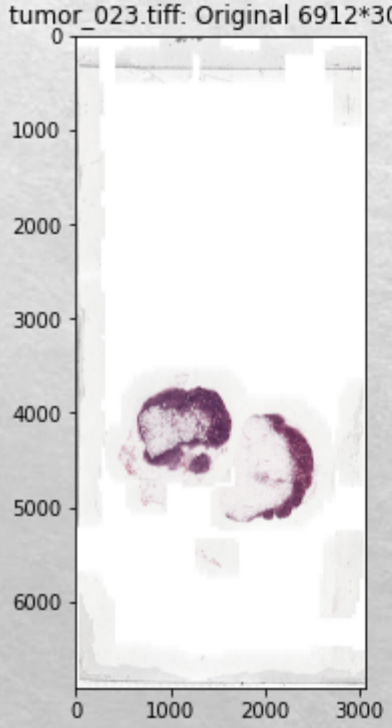
wh2453

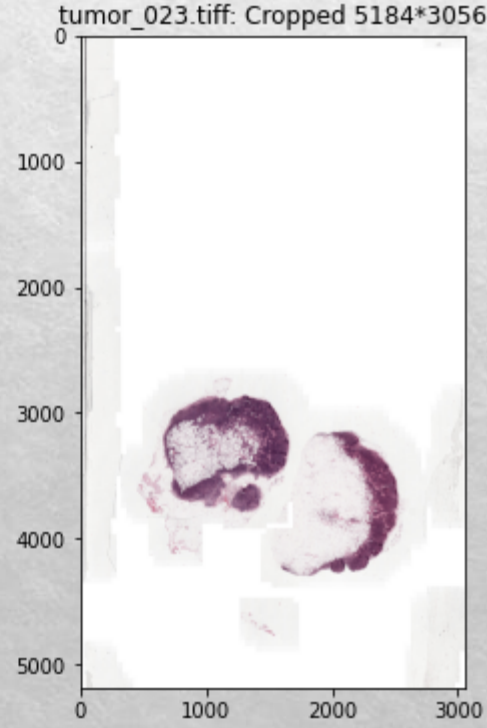
Outline

- Introduction
- Data Processing
- Model Training
- Results & Analysis
- Conclusion


tumor_001.tiff: Original 6912*3056
0
1000
2000
3000
4000
5000
6000
0
1000
2000
3000


tumor_001.tiff: Cropped 5184*3056
0
1000
2000
3000
4000
5000
0
1000
2000
3000


tumor_023.tiff: Original 6912*3056
0
1000
2000
3000
4000
5000
6000
0
1000
2000
3000

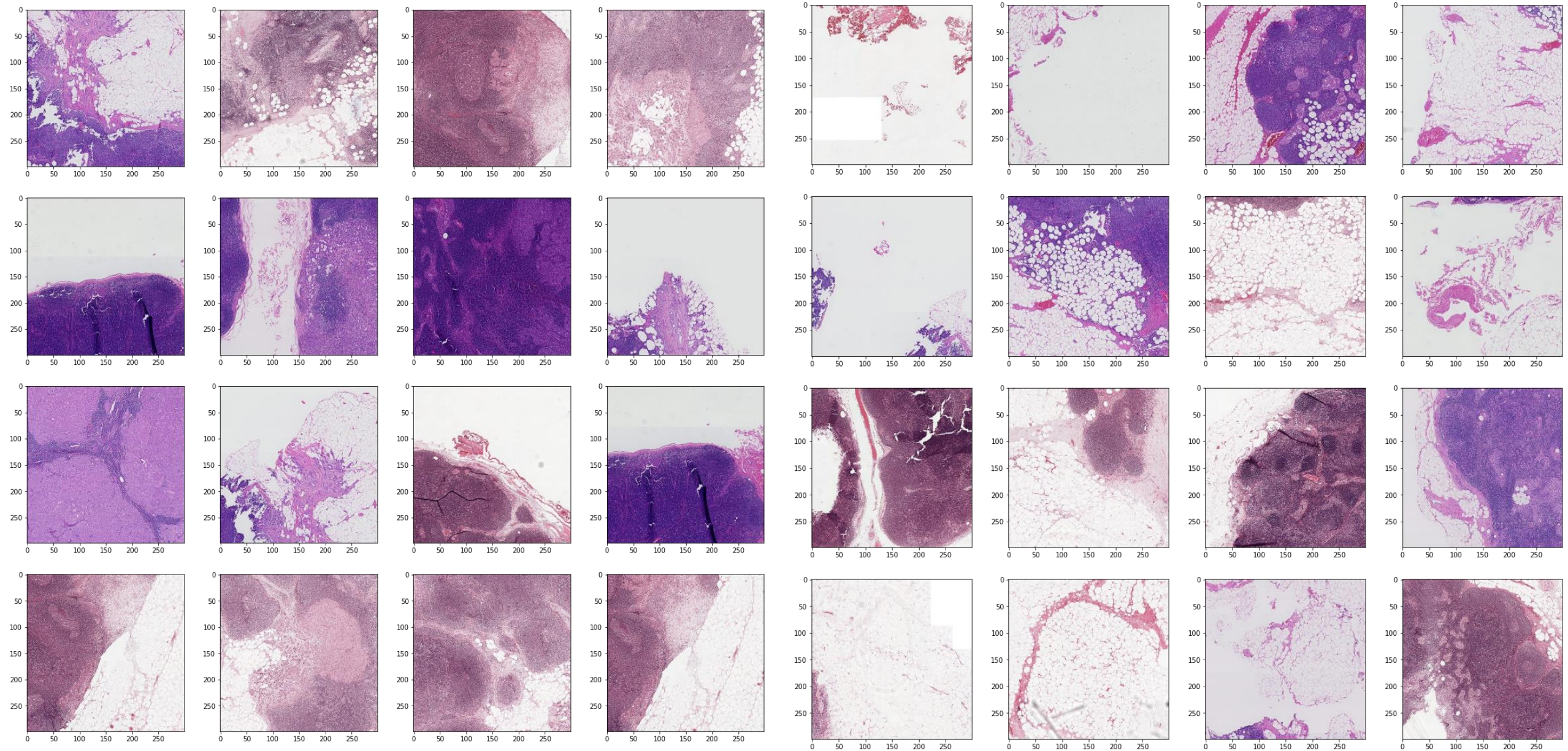

tumor_023.tiff: Cropped 5184*3056
0
1000
2000
3000
4000
5000
0
1000
2000
3000

Generate positive & negative patches

The patch extraction follows two steps:

1. Sliding sample (no overlap): use a slide window of patch size (256*256) to slide over the WSI and crop each region with no overlap

2. Random sample: to avoid in-balanced sample (more normal patches than tumor patches) during the sliding sample process, we further pick random regions to up-sample the patches of the minority class until they have similar number as the majority class.



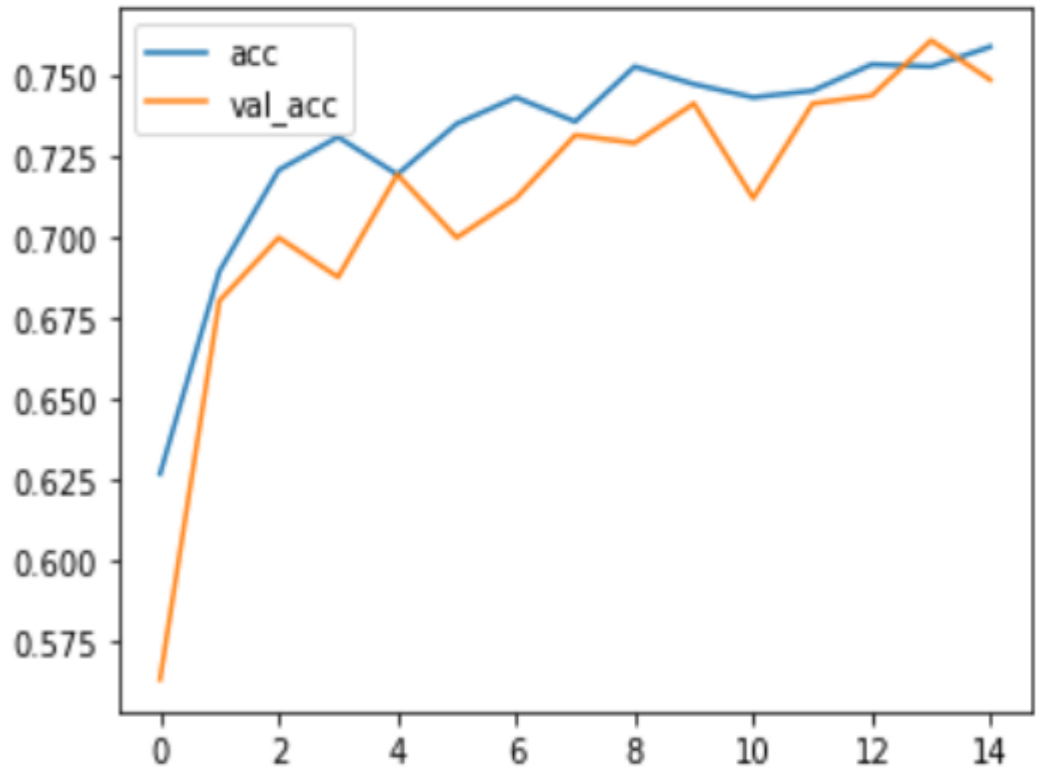
Build Model (VGG16)

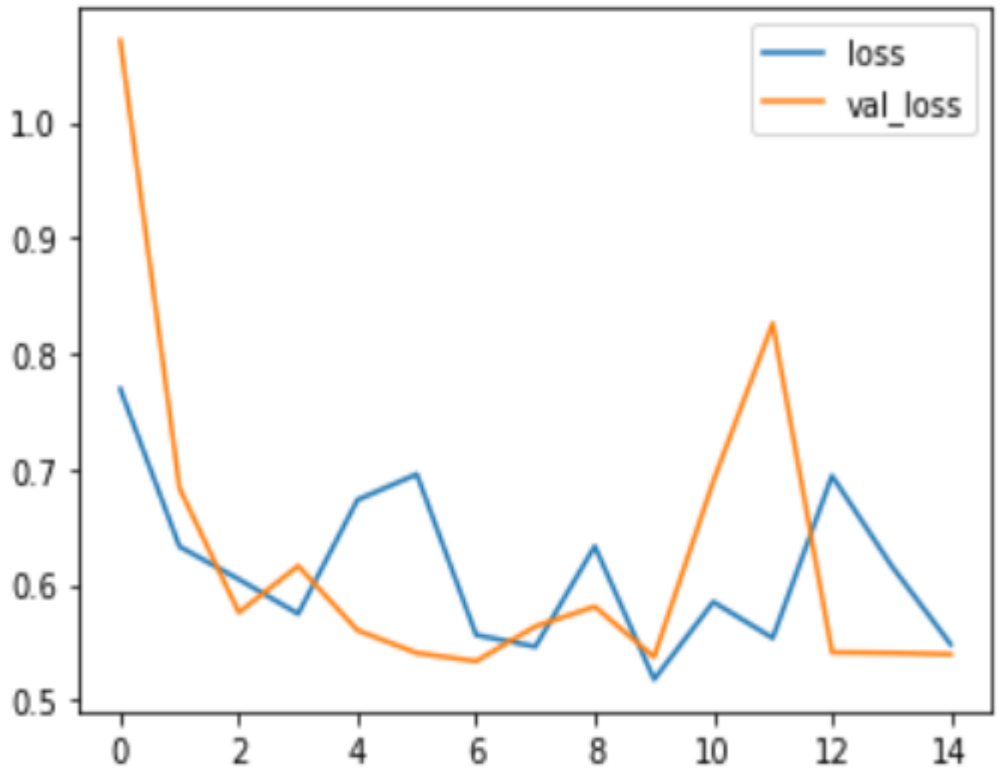
```
[9]  # pre-trained model VGG16
     vgg_conv = VGG16(weights='imagenet',
                      include_top=False,
                      input_shape=(PATCH_SIZE, PATCH_SIZE, 3)) #include_top=False discard the top dense layers
     vgg_conv.trainable = False

     # model 1: level 5 vgg model
     # define new model and add new classifier layer
     vgg_model_5 = Sequential()
     vgg_model_5.add(layers.Input(shape=(PATCH_SIZE, PATCH_SIZE, 3)))
     vgg_model_5.add(vgg_conv)

     vgg_model_5.add(layers.GlobalAveragePooling2D())
     vgg_model_5.add(layers.Dense(256, activation='relu'))
     vgg_model_5.add(layers.Dense(2))

     # summarize
     # vgg_model_5.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 9, 9, 512)         14714688
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 514
=================================================================
Total params: 14,846,530
Trainable params: 131,842
Non-trainable params: 14,714,688
_________________________________________________________________
```

vgg model for level 5 training history

<matplotlib.legend.Legend at 0x7ff26e086b00>





Loss & accuracy

```
[41] test_evaluator_5 = vgg_model_5.evaluate(test_generator_5, steps=len(test_generator_5), return_dict=True)
     print(test_evaluator_5)
```

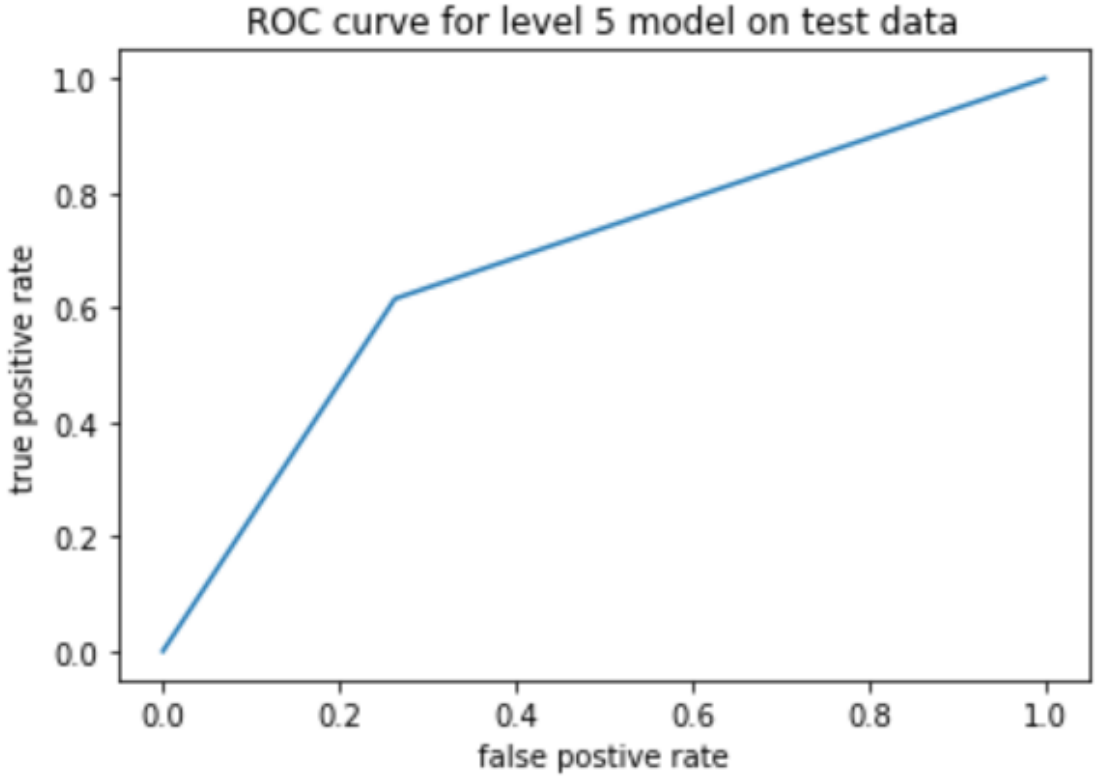
```
13/13 [==============================] - 3s 215ms/step - loss: 0.4012 - acc: 0.7909
{'loss': 0.40116992592811584, 'acc': 0.7909319996833801}
```

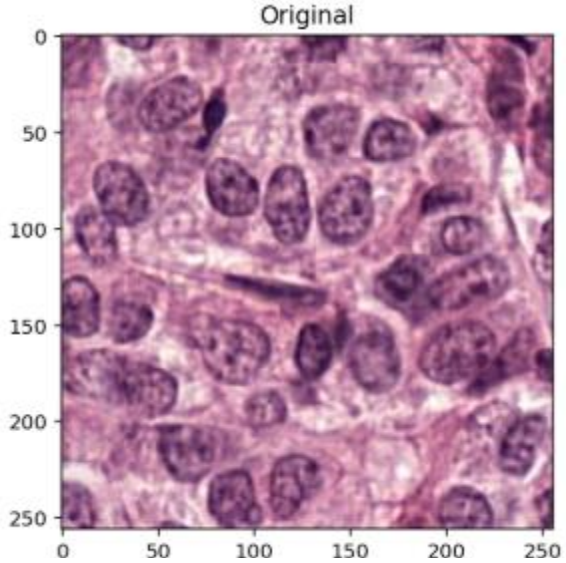
```
[42] test_evaluator_6 = vgg_model_6.evaluate(test_generator_6, steps=len(test_generator_6), return_dict=True)
     print(test_evaluator_6)
```

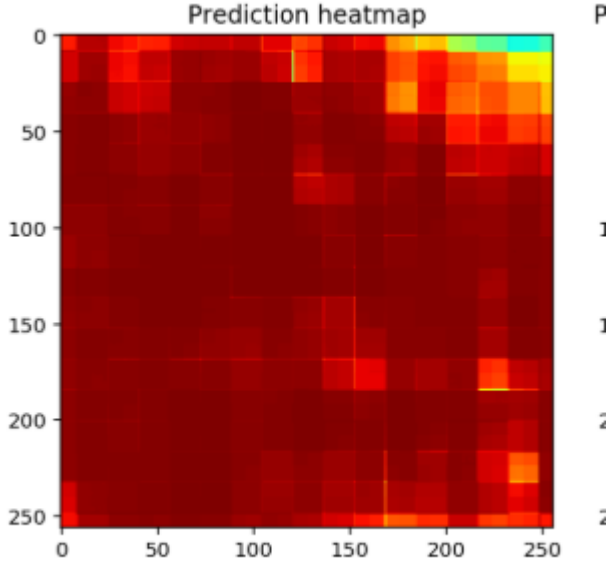
```
5/5 [==============================] - 2s 410ms/step - loss: 4.3841 - acc: 0.6061
{'loss': 4.3841094970703125, 'acc': 0.6060606241226196}
```

```
[135] plt.plot(fpr_array, tpr_array)

      plt.title("ROC curve for level 5 model on test data")
      plt.xlabel("false postive rate")
      plt.ylabel("true positive rate")
```

```
Text(0, 0.5, 'true positive rate')
```




Original


Prediction heatmap