```
!pip install ipython-autotime
!apt-get install openslide-tools
!pip install openslide-python
%load_ext autotime
```

```
Collecting ipython-autotime
  Downloading https://files.pythonhosted.org/packages/b4/c9/b413a24f759641bc27ef
Requirement already satisfied: ipython in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/li
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/py
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dist
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.7/dist
Installing collected packages: ipython-autotime
Successfully installed ipython-autotime-0.3.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  libopenslide0
Suggested packages:
  libtiff-tools
The following NEW packages will be installed:
  libopenslide0 openslide-tools
0 upgraded, 2 newly installed, 0 to remove and 34 not upgraded.
Need to get 92.5 kB of archives.
After this operation, 268 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 libopenslide0 amd64
Get:2 http://archive.ubuntu.com/ubuntu bionic/universe amd64 openslide-tools amd
Fetched 92.5 kB in 1s (118 kB/s)
Selecting previously unselected package libopenslide0.
(Reading database ... 160690 files and directories currently installed.)
Preparing to unpack .../libopenslide0_3.4.1+dfsg-2_amd64.deb ...
Unpacking libopenslide0 (3.4.1+dfsg-2) ...
Selecting previously unselected package openslide-tools.
Preparing to unpack .../openslide-tools_3.4.1+dfsg-2_amd64.deb ...
Unpacking openslide-tools (3.4.1+dfsg-2) ...
Setting up libopenslide0 (3.4.1+dfsg-2) ...
Setting up openslide-tools (3.4.1+dfsg-2) ...
Processing triggers for libc-bin (2.27-3ubuntu1.2) ...
/sbin/ldconfig.real: /usr/local/lib/python3.7/dist-packages/ideep4py/lib/libmkld

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Collecting openslide-python
  Downloading https://files.pythonhosted.org/packages/03/da/12dc0e7566ace61a5a65
     |████████████████████████████████| 317kB 8.3MB/s
```

```
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages
Building wheels for collected packages: openslide-python
  Building wheel for openslide-python (setup.py) ... done
  Created wheel for openslide-python: filename=openslide_python-1.1.2-cp37-cp37m
  Stored in directory: /root/.cache/pip/wheels/6b/55/74/ba9d3dcc2c5c0f1282e08bae
Successfully built openslide-python
Installing collected packages: openslide-python
Successfully installed openslide-python-1.1.2
time: 3.25 ms (started: 2021-04-30 13:02:17 +00:00)
```

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["axes.grid"] = False
import matplotlib.image as mpimg
from openslide import open_slide, __library_version__ as openslide_versi
import seaborn as sns; sns.set_theme()
from sklearn.model_selection import train_test_split
import numpy as np
import pickle
import os
from PIL import Image
```

```
time: 740 ms (started: 2021-04-30 13:02:23 +00:00)
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Globa
from tensorflow.keras.models import Model, Sequential
tf.__version__
```

```
'2.4.1'  time: 1.53 s (started: 2021-04-30 13:02:27 +00:00)
```

```
# mounting the drive
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
time: 16.1 s (started: 2021-04-30 13:02:31 +00:00)
```

```
# utlity function responsible for loading the pickle file from the file_
def load_dataset(file_path):
    with open(file_path, 'rb') as f:
        slides, labels = pickle.load(f)
    return slides, labels
```

```
time: 1.74 ms (started: 2021-04-30 13:02:51 +00:00)
```

Method 1: One Zoom Level

```
PATCH_SIZE = 32
LEVEL = 4
dataset_path = "/content/drive/MyDrive/Columbia_Assignments/ADL/Project/

train_file = os.path.join(dataset_path, 'train/camelyon_preprocessed_lev
test_file = os.path.join(dataset_path, 'test/camelyon_preprocessed_test_
```

```
time: 3.89 ms (started: 2021-04-30 04:01:46 +00:00)
```

```
# load the training dataset and labels

train_slides, train_labels = load_dataset(train_file)
train_slides = np.array(train_slides)
train_labels = np.array(train_labels)
print("Training slides length: {}".format(len(train_slides)))
print("Train labels: {}".format(train_labels))
print("Number of cancerous labels: {}".format(np.sum(train_labels)))
```

```
Training slides length: 146570
Train labels: [0 0 0 ... 0 0 0]
Number of cancerous labels: 8724
time: 8.7 s (started: 2021-04-30 04:01:49 +00:00)
```

```
# load the test dataset and labels

test_slides, test_labels = load_dataset(test_file)
test_slides = np.array(test_slides)
test_labels = np.array(test_labels)
print("Test slides length: {}".format(len(test_slides)))
print("Test labels: {}".format(test_labels))
print("Number of cancerous labels: {}".format(np.sum(test_labels)))
```

```
Test slides length: 63840
Test labels: [0 0 0 ... 0 0 0]
Number of cancerous labels: 5517
time: 3.78 s (started: 2021-04-30 04:02:45 +00:00)
```

Splitting the training dataset into validation and training

```
train_slides, val_slides, train_labels, val_labels = train_test_split(tr
print("Validation slides length: {}".format(len(val_slides)))
print("Validation labels: {}".format(val_labels))
print("Number of cancerous labels: {}".format(np.sum(val_labels)))
```

```
Validation slides length: 36643
Validation labels: [0 0 0 ... 0 0 0]
Number of cancerous labels: 2259
time: 148 ms (started: 2021-04-30 04:02:52 +00:00)
```

Custom Model: Without Data Augmentation

```
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(PAT
model.add(layers.MaxPooling2D())
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D())
model.add(layers.Dropout(0.2))
model.add(layers.Flatten())
model.add(layers.Dense(1, activation='sigmoid'))
```

```
time: 370 ms (started: 2021-04-30 04:02:59 +00:00)
```

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
time: 20 ms (started: 2021-04-30 04:03:03 +00:00)
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 30, 30, 16)        448       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 16)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        9280      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0         
_________________________________________________________________
dropout (Dropout)            (None, 6, 6, 64)          0         
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0         
_________________________________________________________________
dense (Dense)                (None, 1)                 2305      
=================================================================
Total params: 12,033
Trainable params: 12,033
```

```
Non-trainable params: 0
_________________________________________________________________
time: 2.13 ms (started: 2021-04-30 04:03:06 +00:00)
```

Train the model

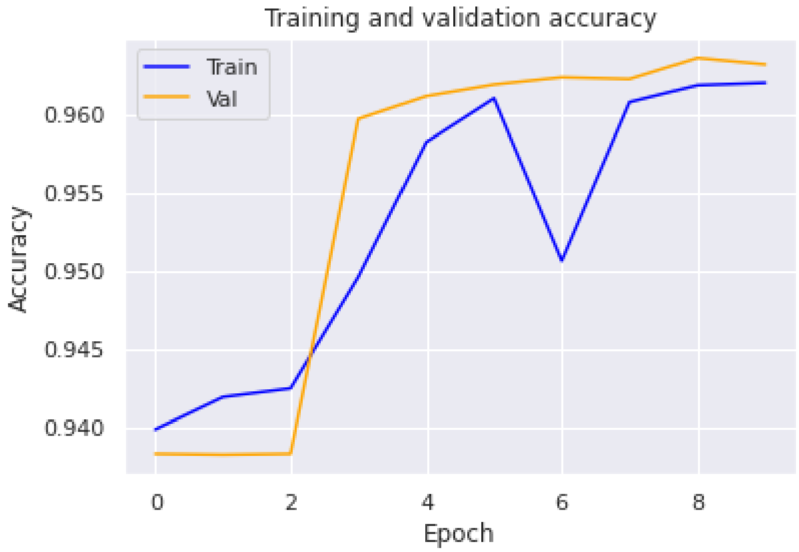
```
epochs = 10
callbacks = [
    keras.callbacks.ModelCheckpoint(os.path.join(dataset_path, 'model/sa
]


history = model.fit(
    train_slides, train_labels,
    validation_data=(val_slides, val_labels),
    epochs=epochs,
    callbacks=callbacks
)
```

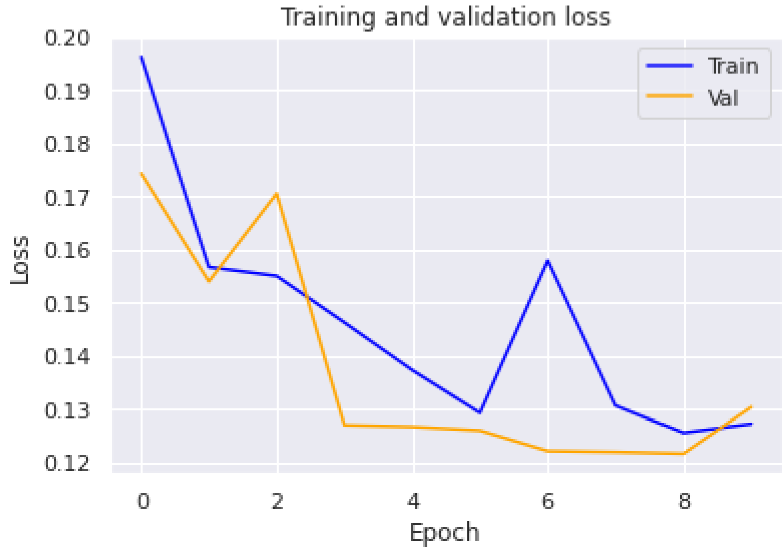
```
==============================] - 78s 23ms/step - loss: 0.3316 - accuracy: 0.9351

==============================] - 77s 22ms/step - loss: 0.1562 - accuracy: 0.9421

==============================] - 77s 22ms/step - loss: 0.1548 - accuracy: 0.9433

==============================] - 79s 23ms/step - loss: 0.1498 - accuracy: 0.9445

==============================] - 77s 22ms/step - loss: 0.1362 - accuracy: 0.9590

==============================] - 78s 23ms/step - loss: 0.1323 - accuracy: 0.9603

==============================] - 76s 22ms/step - loss: 0.1499 - accuracy: 0.9553

==============================] - 77s 23ms/step - loss: 0.1298 - accuracy: 0.9608

==============================] - 78s 23ms/step - loss: 0.1238 - accuracy: 0.9627

==============================] - 78s 23ms/step - loss: 0.1290 - accuracy: 0.9623
23s (started: 2021-04-30 04:04:29 +00:00)
```

```
# Your code here
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

# Get the number of epochs
epochs = range(len(acc))
```

```

plt.title('Training and validation accuracy')
plt.plot(epochs, acc, color='blue', label='Train')
plt.plot(epochs, val_acc, color='orange', label='Val')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

_ = plt.figure()
plt.title('Training and validation loss')
plt.plot(epochs, loss, color='blue', label='Train')
plt.plot(epochs, val_loss, color='orange', label='Val')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f17ddc51ad0>
```





```
time: 745 ms (started: 2021-04-30 04:19:14 +00:00)
```

Test the model on the test data

```python
# load the test tumor tifs
test_tumor_path = os.path.join(dataset_path, 'test/tumor')
test_tumor_mask_path = os.path.join(dataset_path, 'test/tumor_mask')

test_tumors_tifs = []
test_tumors_mask_tifs = []

for filename in os.listdir(test_tumor_path):
    test_tumors_tifs.append(os.path.join(test_tumor_path, filename))
for filename in os.listdir(test_tumor_mask_path):
    test_tumors_mask_tifs.append(os.path.join(test_tumor_mask_path, file

test_tumors_tifs.sort()
test_tumors_mask_tifs.sort()
print("Length of test tumor tiffs: {}".format(len(test_tumors_tifs)))
print("Length of test tumor mask tiffs: {}".format(len(test_tumors_mask_
```

```
Length of test tumor tiffs: 2
Length of test tumor mask tiffs: 2
time: 28.7 ms (started: 2021-04-30 04:19:38 +00:00)
```

```python
# create separate test_slides and test_labels for each of the two test t
# identify the patch index where the first tiff patches end, and the sec
# this is denoted by first_index

stride_width_array = [272, 184]
stride_height_array = [140, 140]

first_index = stride_height_array[0]*stride_width_array[0]
test_slides_normal_1 = test_slides[:first_index]
test_labels_normal_1 = test_labels[:first_index]

test_slides_normal_2 = test_slides[first_index:]
test_labels_normal_2 = test_labels[first_index:]

print("first_tiff_index: ", first_index)
print("length: ", test_labels_normal_1)
print("number of cancerous cells: ", np.sum(test_labels_normal_1))

print("length: ", test_labels_normal_2)
print("number of cancerous cells: ", np.sum(test_labels_normal_2))
```

```
first_tiff_index:  38080
length:  [0 0 0 ... 0 0 0]
number of cancerous cells:  948
```

```
length:  [0 0 0 ... 0 0 0]
number of cancerous cells:  4569
time: 18.2 ms (started: 2021-04-30 04:19:47 +00:00)
```

Predicting the first test tiff file

```
predictions = model.predict(test_slides_normal_1)
```

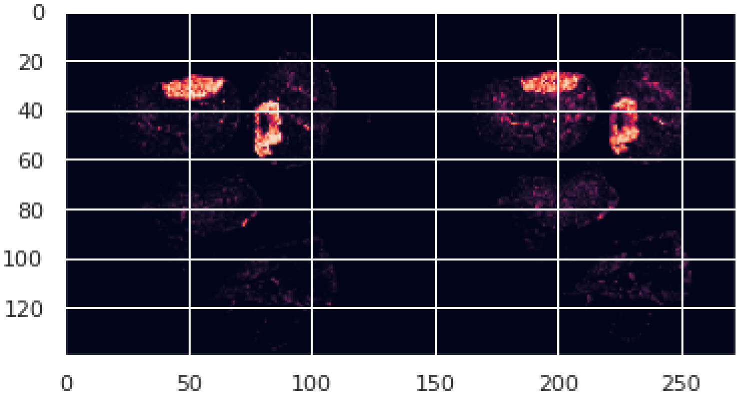
```
time: 8.44 s (started: 2021-04-30 04:19:51 +00:00)
```

```
print("length of predictions: {}".format(len(predictions)))
print("predictions: {}".format(predictions))
```

```
length of predictions: 38080
predictions: [[0.00180888]
 [0.00181362]
 [0.00182429]
 ...
 [0.00170836]
 [0.00163442]
 [0.00162873]]
time: 1.96 ms (started: 2021-04-30 04:20:05 +00:00)
```

```
plt.imshow(np.reshape(predictions, (272, 140)).T)
```

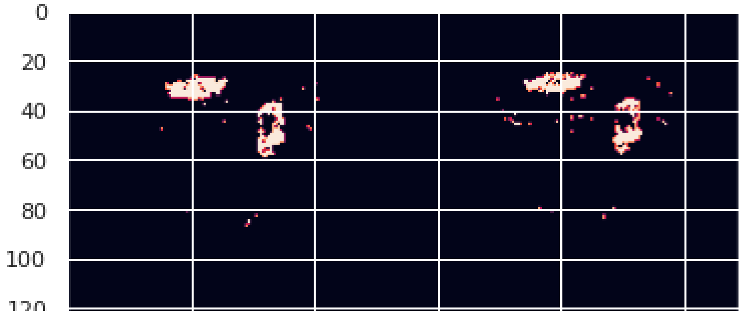
```
<matplotlib.image.AxesImage at 0x7f17dd8da1d0>
```



```
time: 500 ms (started: 2021-04-30 04:20:09 +00:00)
```

```
# temp = np.round(predictions)
THRESHOLD = 0.5
threshold_predictions = np.where(predictions > THRESHOLD, 1, 0)
plt.imshow(np.reshape(threshold_predictions, (272, 140)).T)
```

```
<matplotlib.image.AxesImage at 0x7f17ddd17290>
```



F1-Score

```
time: 320 ms (started: 2021-04-30 04:20:15 +00:00)
```

```
from sklearn.metrics import classification_report

classification_metrics = classification_report(threshold_predictions, te
print(classification_metrics)
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     37481
           1       0.59      0.94      0.73       599

    accuracy                           0.99     38080
   macro avg       0.80      0.96      0.86     38080
weighted avg       0.99      0.99      0.99     38080

time: 53.5 ms (started: 2021-04-30 04:20:19 +00:00)
```

IOU Measure

```
y_pred_image = np.reshape(threshold_predictions, (272, 140))
y_true_image = np.reshape(test_labels_normal_1, (272, 140))

inter = np.logical_and(y_pred_image, y_true_image)
union = np.logical_or(y_pred_image, y_true_image)

print(np.sum(inter)/float(np.sum(union)))
```

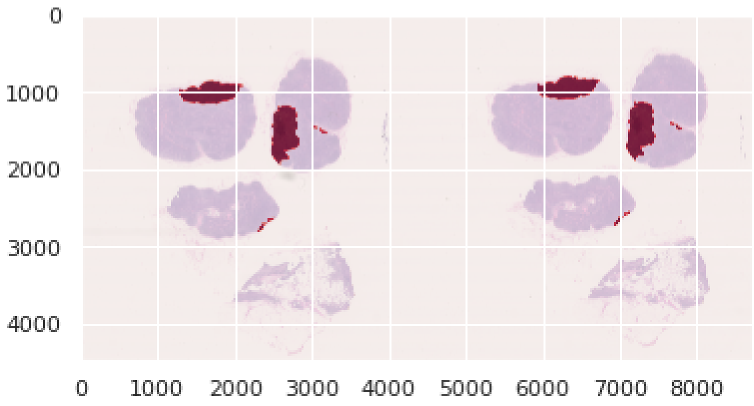
```
0.5705583756345177
time: 8.51 ms (started: 2021-04-30 04:20:31 +00:00)
```

Heatmap for the test tif

```
# read slide and return an image
def read_slide(slide, x, y, level, width, height, as_float=False):
    im = slide.read_region((x,y), level, (width, height))
    im = im.convert('RGB') # drop the alpha channel
    if as_float:
        im = np.asarray(im, dtype=np.float32)
    else:
        im = np.asarray(im)
    assert im.shape == (height, width, 3)
    return im
```

```
time: 7.47 ms (started: 2021-04-30 04:20:38 +00:00)
```

```
# show a slide
tumor_image = open_slide(test_tumors_tifs[0])
mask_image = open_slide(test_tumors_mask_tifs[0])

width = tumor_image.level_dimensions[LEVEL][0]
height = tumor_image.level_dimensions[LEVEL][1]

tumor_slide = read_slide(tumor_image, 0, 0, LEVEL, width=width, height=h
mask_slide = read_slide(mask_image, 0, 0, LEVEL, width=width, height=he

plt.imshow(tumor_slide)
plt.imshow(mask_slide[:, :, 0], cmap='Reds', alpha=0.7)
plt.show()
```

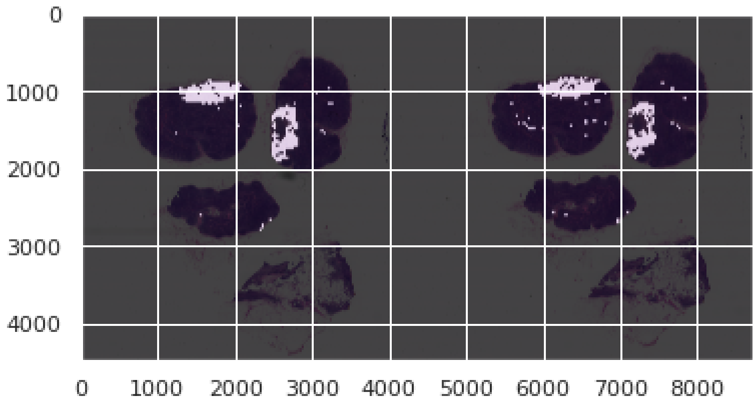


```
time: 8.04 s (started: 2021-04-30 04:20:43 +00:00)
```

```
the patches to see if patch extraction is working properly
dth = width // PATCH_SIZE
ight = height // PATCH_SIZE

slide = Image.new('RGB', (PATCH_SIZE * step_width, PATCH_SIZE * step_hei
nask = Image.new('RGB', (PATCH_SIZE * step_width, PATCH_SIZE * step_heig

ank = np.zeros((PATCH_SIZE, PATCH_SIZE))
nor = 255 * np.ones((PATCH_SIZE, PATCH_SIZE))
```

```
 0

n range(step_width):
 j in range(step_height):
 canvas_slide.paste(Image.fromarray(test_slides_normal_1[index], 'RGB'),
 if threshold_predictions[index] == 0:
     canvas_mask.paste(Image.fromarray(mask_blank), (i*PATCH_SIZE, j*PAT
 else:
     canvas_mask.paste(Image.fromarray(mask_tumor), (i*PATCH_SIZE, j*PAT
 index += 1

ame = 'patch_to_slide.png'
me = 'patch_to_mask.png'
slide.save(slide_name)
mask.save(mask_name)
lide = plt.imread(slide_name)
ask = plt.imread(mask_name)

how(image_slide)
how(image_mask, cmap='jet', alpha=0.7)
```

```
<matplotlib.image.AxesImage at 0x7f17dd7b8d50>
```



```
time: 22.8 s (started: 2021-04-30 04:21:27 +00:00)
```

Predicting the second tiff file

```
predictions_2 = model.predict(test_slides_normal_2)
```

```
time: 5.66 s (started: 2021-04-30 04:21:57 +00:00)
```

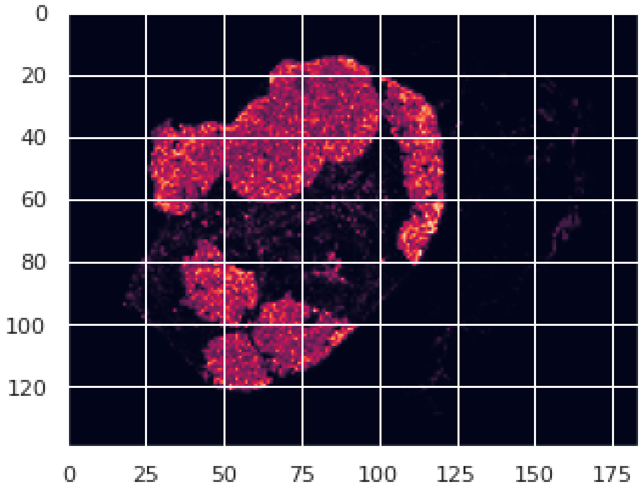
```
print("length of predictions: {}".format(len(predictions_2)))
print("predictions: {}".format(predictions_2))
```

```
length of predictions: 25760
predictions: [[0.00125578]
```

```
 [0.00113297]
 [0.00137514]
 ...
 [0.00104704]
 [0.00114307]
 [0.00122961]]
time: 2.18 ms (started: 2021-04-30 04:22:08 +00:00)
```

```
plt.imshow(np.reshape(predictions_2, (184, 140)).T)
```

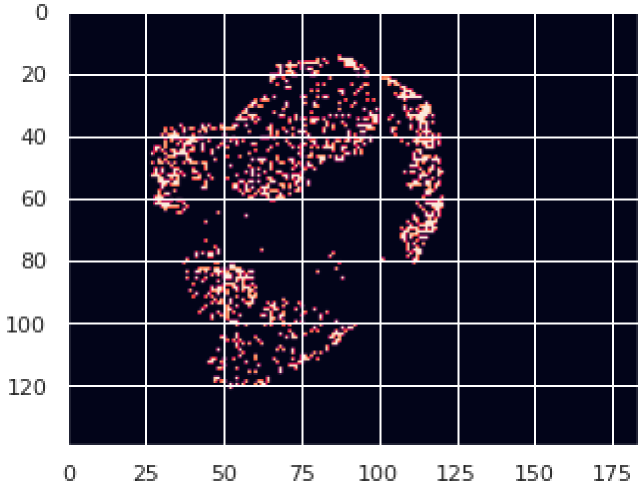
```
<matplotlib.image.AxesImage at 0x7f17db6b1a90>
```



```
time: 341 ms (started: 2021-04-30 04:22:11 +00:00)
```

```
# temp = np.round(predictions)
THRESHOLD = 0.5
threshold_predictions = np.where(predictions_2 > THRESHOLD, 1, 0)
plt.imshow(np.reshape(threshold_predictions, (184, 140)).T)
```

```
<matplotlib.image.AxesImage at 0x7f17db67a590>
```



```
time: 324 ms (started: 2021-04-30 04:22:15 +00:00)
```

```
from sklearn.metrics import classification_report

```

```
classification_metrics = classification_report(threshold_predictions, te
print(classification_metrics)
```

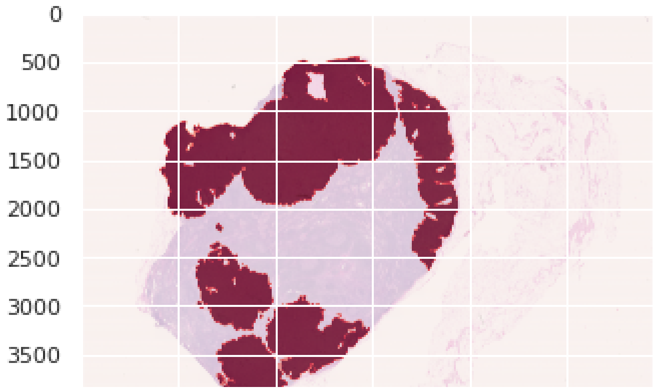
```
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     24649
           1       0.24      0.99      0.39      1111

    accuracy                           0.86     25760
   macro avg       0.62      0.92      0.66     25760
weighted avg       0.97      0.86      0.90     25760

time: 35.5 ms (started: 2021-04-30 04:22:20 +00:00)
```

Heatmap for the test file

```
# read slide and return an image
def read_slide(slide, x, y, level, width, height, as_float=False):
    im = slide.read_region((x,y), level, (width, height))
    im = im.convert('RGB') # drop the alpha channel
    if as_float:
        im = np.asarray(im, dtype=np.float32)
    else:
        im = np.asarray(im)
    assert im.shape == (height, width, 3)
    return im
```

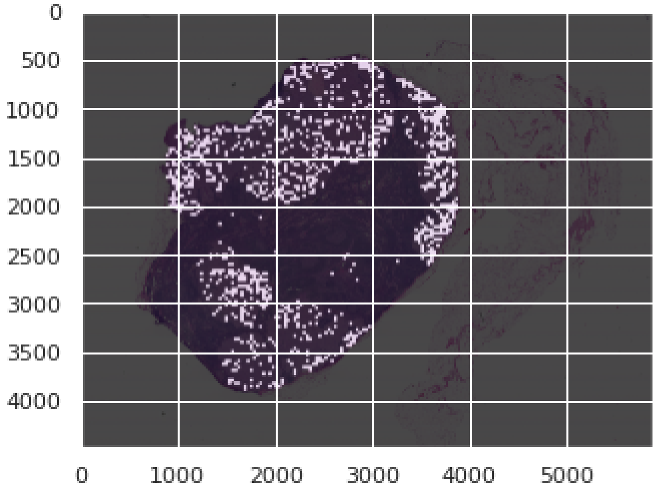
```
time: 7.8 ms (started: 2021-04-30 04:22:23 +00:00)
```

```
# show a slide
tumor_image = open_slide(test_tumors_tifs[1])
mask_image = open_slide(test_tumors_mask_tifs[1])

width = tumor_image.level_dimensions[LEVEL][0]
height = tumor_image.level_dimensions[LEVEL][1]

tumor_slide = read_slide(tumor_image, 0, 0, LEVEL, width=width, height=h
mask_slide = read_slide(mask_image, 0, 0, LEVEL, width=width, height=he

plt.imshow(tumor_slide)
plt.imshow(mask_slide[:, :, 0], cmap='Reds', alpha=0.7)
plt.show()
```



```
 plot the patches to see if patch extraction is working properly
tep_width = width // PATCH_SIZE
tep_height = height // PATCH_SIZE
anvas_slide = Image.new('RGB', (PATCH_SIZE * step_width, PATCH_SIZE * st
anvas_mask = Image.new('RGB', (PATCH_SIZE * step_width, PATCH_SIZE * ste
ask_blank = np.zeros((PATCH_SIZE, PATCH_SIZE))
ask_tumor = 255 * np.ones((PATCH_SIZE, PATCH_SIZE))
ndex = 0

or i in range(step_width):
   for j in range(step_height):
       canvas_slide.paste(Image.fromarray(test_slides_normal_2[index], '
       if threshold_predictions[index] == 0:
           canvas_mask.paste(Image.fromarray(mask_blank), (i*PATCH_SIZE,
       else:
           canvas_mask.paste(Image.fromarray(mask_tumor), (i*PATCH_SIZE,
       index += 1

lide_name = 'patch_to_slide.png'
ask_name = 'patch_to_mask.png'
anvas_slide.save(slide_name)
anvas_mask.save(mask_name)
mage_slide = plt.imread(slide_name)
mage_mask = plt.imread(mask_name)

lt.imshow(image_slide)
lt.imshow(image_mask, cmap='Reds', alpha=0.7)
```

```
<matplotlib.image.AxesImage at 0x7f17d5a4c150>
```



```
time: 16.1 s (started: 2021-04-30 04:23:40 +00:00)
```

✓ 10s completed at 9:03 AM