Name: Anik Manik

Email address: iamanik4@gmail.com

Contact number: 9477672426

Anydesk address: 400 728 410

Years of Work Experience: 2.6 years

Date: 24th Jan 2021

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from PIL import Image
import cv2
import glob
import glob2
%matplotlib inline
from tqdm import tqdm
import shutil
```

In [2]:
```
!pip install pydicom
import pydicom
```

```
Collecting pydicom
  Downloading https://files.pythonhosted.org/packages/f4/15/df16546bc59bfca390cf072d473fb2c8acd423163
6f64356593a63137e55/pydicom-2.1.2-py3-none-any.whl (1.9MB)
     |████████████████████████████████| 1.9MB 5.3MB/s
Installing collected packages: pydicom
Successfully installed pydicom-2.1.2
```

https://www.kaggle.com/seesee/siim-train-test (https://www.kaggle.com/seesee/siim-train-test)

In [3]:
```
# download the dataset
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win6
4; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.190 Safari/537.36" --header="Accept: t
ext/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: http
s://www.kaggle.com/" --header="Cookie: ext_name=ojplmecpdpgccookcobabopnaifgidhf" --header="Connectio
n: keep-alive" "https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archive.zip?X-Goo
g-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%
2F20210301%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210301T175422Z&X-Goog-Expires=259199&X-Goog-
SignedHeaders=host&X-Goog-Signature=94f80fbe22ed7bfd9b73e2f1a7c3d3cd306876aad5e95b6f699a868f46bc29114b
d843ba6727a13fc2666efcd8382ade7c0e11324c428b825fc1b23b8038546cb7902c2b4e43de14675f4df53027c864d84714db
8e7d2cad90b8a038a057e177843afcbc538e6f99f9cea11833be543b9b3c6a3f2ba969ab11d41db613a8a83909f7042a4c91dc
c6b1e8cc764f24ba78c8ce943e7f34bcaaf82bc0dbe416661429e5000ed0bcfe9034c0abe07918a47c100dd6551275770f9499
5eab922a62e4a0a8fddd6e353bc3887d03bf81fc18fcaea66bc70eb0dac9cf7af9d3e75e311d1505c54c686ea0eb891d1b94f1
8446cddedc9ee0e94f81942e668efc2041faaa" -c -O 'archive.zip'
```

```
--2021-03-02 03:07:45--  https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archiv
e.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gservice
account.com%2F20210301%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210301T175422Z&X-Goog-Expires=2
59199&X-Goog-SignedHeaders=host&X-Goog-Signature=94f80fbe22ed7bfd9b73e2f1a7c3d3cd306876aad5e95b6f699a
868f46bc29114bd843ba6727a13fc2666efcd8382ade7c0e11324c428b825fc1b23b8038546cb7902c2b4e43de14675f4df53
027c864d84714db8e7d2cad90b8a038a057e177843afcbc538e6f99f9cea11833be543b9b3c6a3f2ba969ab11d41db613a8a8
3909f7042a4c91dcc6b1e8cc764f24ba78c8ce943e7f34bcaaf82bc0dbe416661429e5000ed0bcfe9034c0abe07918a47c100
dd6551275770f94995eab922a62e4a0a8fddd6e353bc3887d03bf81fc18fcaea66bc70eb0dac9cf7af9d3e75e311d1505c54c
686ea0eb891d1b94f18446cddedc9ee0e94f81942e668efc2041faaa
Resolving storage.googleapis.com (storage.googleapis.com)... 108.177.125.128, 74.125.23.128, 64.233.1
89.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|108.177.125.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2059765561 (1.9G) [application/zip]
Saving to: ‘archive.zip’

archive.zip         100%[===================>]   1.92G  55.1MB/s    in 56s

2021-03-02 03:08:42 (34.9 MB/s) - ‘archive.zip’ saved [2059765561/2059765561]
```

In [4]:
```
# unzip the data
!unzip -qq 'archive.zip'
```

In [5]:
```
# read the given train csv file
train_df = pd.read_csv('siim/train-rle.csv')
train_df.head()
```

Out[5]:

	ImageId	EncodedPixels
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...

This csv file contains the ImageID and RLE encoded masks.

In [6]:
```
# check the properties of the train dataframe
train_df.describe()
```

Out[6]:

	ImageId	EncodedPixels
count	12954	12954
unique	12047	3577
top	1.2.276.0.7230010.3.1.4.8323329.1851.151787516...	-1
freq	10	9378

Out of 12954 imageID, 12047 are unique. It means there are duplicates.

In [7]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12954 entries, 0 to 12953
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   ImageId         12954 non-null  object
 1    EncodedPixels  12954 non-null  object
dtypes: object(2)
memory usage: 202.5+ KB
```

There are no null values in this dataset.

Drop the duplicate ImageIDs

In [8]:
```
# drop the duplicate ImageIDs
train_df.drop_duplicates(subset ="ImageId", keep = 'first', inplace = True)
```

In [9]:
```
train_df.describe()
```

Out[9]:

	ImageId	EncodedPixels
count	12047	12047
unique	12047	2670
top	1.2.276.0.7230010.3.1.4.8323329.2029.151787517...	-1
freq	1	9378

Now there are no duplicates left in the dataset, all 12047 ImageIDs are unique.

In [10]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12047 entries, 0 to 12953
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   ImageId         12047 non-null  object
 1    EncodedPixels  12047 non-null  object
dtypes: object(2)
memory usage: 282.4+ KB
```

Create 2 separate directory to store Train and Test images

In [11]:
```
from tqdm import tqdm
import shutil

# create 2 separate directories for train and test dicom images
train_images_dicom = 'siim/train_images_dicom/'
test_images_dicom = 'siim/test_images_dicom/'

if not os.path.isdir(train_images_dicom):
    os.makedirs(train_images_dicom)
if not os.path.isdir(test_images_dicom):
    os.makedirs(test_images_dicom)
```

Move the images to respective directory

In [12]:
```
# move all train dicom images from 'dicom-images-train' to 'train_images_dicom' in a single directory
train_path = 'siim/dicom-images-train/'
train_dcm_list = glob2.glob(os.path.join(train_path, '**/*.dcm'))
for filename in tqdm(train_dcm_list):
  shutil.move(str(filename), train_images_dicom)

# move all test dicom images from 'dicom-images-test' to 'test_images_dicom' in a single directory
test_path = 'siim/dicom-images-test/'
test_dcm_list = glob2.glob(os.path.join(test_path, '**/*.dcm'))
for filename in tqdm(test_dcm_list):
  shutil.move(str(filename), test_images_dicom)
```

```
100%|██████████| 12089/12089 [00:00<00:00, 20788.75it/s]
100%|██████████| 3205/3205 [00:00<00:00, 24000.83it/s]
```

Create a dataframe containing ImageId, EncodedPixels and ImagePath

In [13]:
```
# create a dataframe containing ImageId, EncodedPixels and ImagePath
# dicom images has '.dcm' extension, concating '.dcm' with ImageId I get filename
# train_df['DICOM_Path'] = os.path.join(train_images_dicom, (train_df['ImageId']+'.dcm'))
train_df['Image_Path'] = train_images_dicom + train_df['ImageId']+'.dcm'
train_df.head()
```

Out[13]:

	ImageId	EncodedPixels	Image_Path
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...	siim/train_images_dicom/1.2.276.0.7230010.3.1....

In [14]:
```
# create test dataframe from the test ImageIDs given
test_df = pd.DataFrame(columns = ['ImageId', 'Image_Path'])
test_data = os.listdir(test_images_dicom)
for file in test_data:
  test_df = test_df.append({'ImageId':file[:-4], 'Image_Path':test_images_dicom+file}, ignore_index =
True)
test_df.head()
```

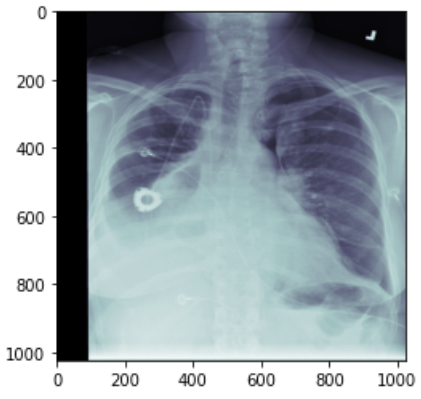
Out[14]:

	ImageId	Image_Path
0	ID_81f0c8200	siim/test_images_dicom/ID_81f0c8200.dcm
1	ID_6124ce73f	siim/test_images_dicom/ID_6124ce73f.dcm
2	ID_b4427a13c	siim/test_images_dicom/ID_b4427a13c.dcm
3	ID_e1af81d4e	siim/test_images_dicom/ID_e1af81d4e.dcm
4	ID_5cd49901c	siim/test_images_dicom/ID_5cd49901c.dcm

Display a sample image

In [15]:
```
# displaying a sample image
# https://stackoverflow.com/questions/48185544/read-and-open-dicom-images-using-python
image_row = 4
sample_img = pydicom.read_file(train_df['Image_Path'][image_row]).pixel_array
print('ImageId = '+str(train_df['ImageId'][image_row]))
plt.imshow(sample_img, cmap='bone')
plt.show()
```

```
ImageId = 1.2.276.0.7230010.3.1.4.8323329.10016.1517875220.992175
```



In [16]:
```
# remove extra space in EncodedPixels column
train_df.rename(columns = {' EncodedPixels':'EncodedPixels'}, inplace = True)
```

In [17]:
```
# add a column whether the image is with pneumothorax or without pneumothorax
# train_df['Whether_Pneumothorax'] = np.where(train_df['EncodedPixels']=='-1', 'No', 'Yes')
train_df['Whether_Pneumothorax'] = np.where(train_df['EncodedPixels']=='-1', 0, 1)
train_df.head(10)
```

Out[17]:

	ImageId	EncodedPixels	Image_Path
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
5	1.2.276.0.7230010.3.1.4.8323329.11444.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
6	1.2.276.0.7230010.3.1.4.8323329.32219.15178751...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
7	1.2.276.0.7230010.3.1.4.8323329.32395.15178751...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
8	1.2.276.0.7230010.3.1.4.8323329.10623.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
9	1.2.276.0.7230010.3.1.4.8323329.12095.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....

In [18]:
```
# save the .csv file for further use
train_df.to_csv('train_dicom_imageid_path.csv', index=False)
```

create 2 separate directories for train and test png images

In [19]:
```
# create 2 separate directories for train and test png images
train_images_png = 'siim/train_images_png/'
test_images_png = 'siim/test_images_png/'

if not os.path.isdir(train_images_png):
    os.makedirs(train_images_png)
if not os.path.isdir(test_images_png):
    os.makedirs(test_images_png)
```

Define function to convert dicom images to png

In [20]:
```
# Define function to convert dicom images to png
def convert_images_to_png(filename, outdir):
  ds = pydicom.read_file(str(filename))
  img = ds.pixel_array
  img = cv2.resize(img, (256, 256))
  cv2.imwrite(outdir + filename.split('/')[-1][:-4] + '.png', img)
```

In [21]:
```
# convert train images
for file in tqdm(train_df['Image_Path']):
  convert_images_to_png(file, train_images_png)

# convert test images
for file in tqdm(test_df['Image_Path']):
  convert_images_to_png(file, test_images_png)
```

```
100%|██████████| 12047/12047 [02:03<00:00, 97.70it/s]
100%|██████████| 3205/3205 [00:33<00:00, 96.60it/s]
```

In [22]:
```
# Define function to convert RLE to mask, provided by organizers
def rle2mask(rle, width, height):
    mask= np.zeros(width* height)
    array = np.asarray([int(x) for x in rle.split()])
    starts = array[0::2]
    lengths = array[1::2]

    current_position = 0
    for index, start in enumerate(starts):
        current_position += start
        mask[current_position:current_position+lengths[index]] = 1
        current_position += lengths[index]

    return mask.reshape(width, height)
```

Create Directories for mask png files

In [23]:
```
# Create Directories for mask png files
train_mask_png = 'siim/train_mask_png/'

if not os.path.isdir(train_mask_png):
    os.makedirs(train_mask_png)
```

In [24]:
```
def convert_images_to_png(filename, outdir):
  ds = pydicom.read_file(str(filename))
  img = ds.pixel_array
  img = cv2.resize(img, (256, 256))
  cv2.imwrite(outdir + filename.split('/')[-1][:-4] + '.png', img)
```

In [25]:
```
# define function to convert mask to png image
def convert_masks_to_png(data, outdir):
  for img_id, enc_pix in tqdm(data.values):
    mask_path = outdir + str(img_id) + '_mask.png'
    # print(mask_path)
    if enc_pix != "-1":
      image_bytes = rle2mask(enc_pix, 1024, 1024).T
      mask = cv2.resize(image_bytes, (256, 256))
      cv2.imwrite(mask_path, mask)
    else:
      mask = np.zeros((256, 256), dtype=np.uint8)
      cv2.imwrite(mask_path, mask)
convert_masks_to_png(train_df[['ImageId', 'EncodedPixels']], train_mask_png)
```

```
100%|██████████| 12047/12047 [00:28<00:00, 426.63it/s]
```

create a dataframe containing full image path and mask path

In [26]:
```
# create a dataframe containing full image path and mask path
train_image_mask_df = pd.DataFrame(columns = ['ImageId', 'Image_Path', 'Mask_Path'])
for img_id in tqdm(train_df['ImageId']):
  # print(img_id)
  image_path = train_images_png + str(img_id) + '.png'
  mask_path = train_mask_png + str(img_id) + '_mask.png'
  # print(image_path)
  # print(mask_path)
  train_image_mask_df = train_image_mask_df.append({'ImageId':img_id, 'Image_Path':image_path, 'Mask_P
ath':mask_path}, ignore_index = True)

train_image_mask_df.head()
```

```
100%|██████████| 12047/12047 [00:32<00:00, 372.26it/s]
```

Out[26]:

	ImageId	Image_Path	
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_

In [27]:
```
# create a dataframe containing full image path and mask path
train_image_mask_df = pd.DataFrame(columns = ['ImageId', 'Image_Path', 'Mask_Path', 'is_pneumothorax'
])
for i in tqdm(range(len(train_df))):
  img_id = train_df['ImageId'].iloc[i]
  is_ptx = train_df['Whether_Pneumothorax'].iloc[i]
  # print(img_id)
  image_path = train_images_png + str(img_id) + '.png'
  mask_path = train_mask_png + str(img_id) + '_mask.png'
  # print(image_path)
  # print(mask_path)
  train_image_mask_df = train_image_mask_df.append({'ImageId':img_id, 'Image_Path':image_path, 'Mask_P
ath':mask_path, 'is_pneumothorax':is_ptx}, ignore_index = True)

train_image_mask_df.head()
```

```
100%|██████████| 12047/12047 [00:46<00:00, 258.46it/s]
```

Out[27]:

	ImageId	Image_Path	
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_

In [28]:
```
train_image_mask_df.head(20)
```

Out[28]:

	ImageId	Image_Path	
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
5	1.2.276.0.7230010.3.1.4.8323329.11444.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
6	1.2.276.0.7230010.3.1.4.8323329.32219.15178751...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
7	1.2.276.0.7230010.3.1.4.8323329.32395.15178751...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
8	1.2.276.0.7230010.3.1.4.8323329.10623.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
9	1.2.276.0.7230010.3.1.4.8323329.12095.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
10	1.2.276.0.7230010.3.1.4.8323329.3514.151787517...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
11	1.2.276.0.7230010.3.1.4.8323329.3297.151787517...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
12	1.2.276.0.7230010.3.1.4.8323329.10930.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
13	1.2.276.0.7230010.3.1.4.8323329.14008.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
14	1.2.276.0.7230010.3.1.4.8323329.12017.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
15	1.2.276.0.7230010.3.1.4.8323329.6340.151787519...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
16	1.2.276.0.7230010.3.1.4.8323329.11977.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
17	1.2.276.0.7230010.3.1.4.8323329.4965.151787518...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
18	1.2.276.0.7230010.3.1.4.8323329.11405.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask
19	1.2.276.0.7230010.3.1.4.8323329.3767.151787517...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask

In [29]:
```
test_image_df = pd.DataFrame(columns = ['ImageId', 'Image_Path'])
for img_id in tqdm(test_df['ImageId']):
  image_path = test_images_png + str(img_id) + '.png'
  test_image_df = test_image_df.append({'ImageId':img_id, 'Image_Path':image_path}, ignore_index = Tru
e)

test_image_df.head()
```

```
100%|██████████| 3205/3205 [00:07<00:00, 433.41it/s]
```

Out[29]:

	ImageId	Image_Path
0	ID_81f0c8200	siim/test_images_png/ID_81f0c8200.png
1	ID_6124ce73f	siim/test_images_png/ID_6124ce73f.png
2	ID_b4427a13c	siim/test_images_png/ID_b4427a13c.png
3	ID_e1af81d4e	siim/test_images_png/ID_e1af81d4e.png
4	ID_5cd49901c	siim/test_images_png/ID_5cd49901c.png

Build Data Pipeline

In [30]:
```
train_image_mask_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12047 entries, 0 to 12046
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   ImageId          12047 non-null  object
 1   Image_Path       12047 non-null  object
 2   Mask_Path        12047 non-null  object
 3   is_pneumothorax  12047 non-null  object
dtypes: object(4)
memory usage: 376.6+ KB
```

In [31]:
```
train_image_mask_df.is_pneumothorax.value_counts()
```

Out[31]:
```
0    9378
1    2669
Name: is_pneumothorax, dtype: int64
```

In [32]:
```
train_df['Whether_Pneumothorax'].value_counts()
```

Out[32]:
```
0    9378
1    2669
Name: Whether_Pneumothorax, dtype: int64
```

In [33]:
```
from sklearn.model_selection import train_test_split
x_train, x_val = train_test_split(train_image_mask_df[['Image_Path', 'Mask_Path']], stratify=train_ima
ge_mask_df.is_pneumothorax, test_size=0.15, random_state=42)

print(x_train.shape)
print(x_val.shape)
```

```
(10239, 2)
(1808, 2)
```

In [73]:
```
from glob import glob
import shutil
import argparse
import zipfile
import hashlib
import requests
from tqdm import tqdm
import IPython.display as display
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import datetime, os
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import Adam
from IPython.display import clear_output
# import tensorflow_addons as tfa

# For more information about autotune:
# https://www.tensorflow.org/guide/data_performance#prefetching
AUTOTUNE = tf.data.experimental.AUTOTUNE
print(f"Tensorflow ver. {tf.__version__}")
```

```
Tensorflow ver. 2.4.1
```

In [74]:
```
def read_decode_image(image_path):
  # image_path = image_mask_path['image_path']
  # mask_path = image_mask_path['mask_path']
  image = tf.io.read_file(image_path)
  image = tf.image.decode_png(image, channels=3)
  image = tf.image.convert_image_dtype(image, tf.float32)
  return image

def read_decode_mask(mask_path):
  mask = tf.io.read_file(mask_path)
  mask = tf.image.decode_png(mask, channels=1)
  mask = tf.image.convert_image_dtype(mask, tf.float32)
  # return {'image_file': image, 'mask_file': mask}
  return mask
```

In [75]:
```
SEED = 42
train_image = tf.data.Dataset.list_files(x_train['Image_Path'], seed=SEED)
train_image = train_image.map(read_decode_image)

train_mask = tf.data.Dataset.list_files(x_train['Mask_Path'], seed=SEED)
train_mask = train_mask.map(read_decode_mask)

train_dataset = (train_image, train_mask)

val_image = tf.data.Dataset.list_files(x_train['Image_Path'], seed=SEED)
val_image = val_image.map(read_decode_image)

val_mask = tf.data.Dataset.list_files(x_val['Mask_Path'], seed=SEED)
val_mask = val_mask.map(read_decode_mask)

val_dataset = (val_image, val_mask)
```

In [76]:
```
train_dataset, val_dataset
```

Out[76]:
```
((<MapDataset shapes: (None, None, 3), types: tf.float32>,
  <MapDataset shapes: (None, None, 1), types: tf.float32>),
 (<MapDataset shapes: (None, None, 3), types: tf.float32>,
  <MapDataset shapes: (None, None, 1), types: tf.float32>))
```

In [77]:
```
def load_image_train(x, y):
  input_image = tf.image.resize(x, (IMG_SIZE, IMG_SIZE))
  input_mask = tf.image.resize(y, (IMG_SIZE, IMG_SIZE))

  # apply augmentation in train daya only
  random_value = tf.random.uniform(())
  if random_value > 0.5:
    input_image = tf.image.flip_left_right(input_image)
    input_mask = tf.image.flip_left_right(input_mask)

  return input_image, input_mask

def load_image_val(x, y):
  input_image = tf.image.resize(x, (IMG_SIZE, IMG_SIZE))
  input_mask = tf.image.resize(y, (IMG_SIZE, IMG_SIZE))

  return input_image, input_mask
```

In [78]:
```
def preprocess_image_mask(x, y):
  decoded_image = read_decode_image(x)
  decoded_mask = read_decode_mask(y)

  return decoded_image, decoded_mask
```

In [79]:
```
BATCH_SIZE = 5
BUFFER_SIZE = 1000
IMG_SIZE = 256
SEED = 42

x = x_train['Image_Path']
y = x_train['Mask_Path']
# Train Dataset
dataset_train = tf.data.Dataset.from_tensor_slices((x, y))
dataset_train = dataset_train.map(lambda x, y: preprocess_image_mask(x, y), num_parallel_calls=tf.data
.experimental.AUTOTUNE)
dataset_train = dataset_train.map(lambda x, y: load_image_train(x, y), num_parallel_calls=tf.data.expe
rimental.AUTOTUNE)
dataset_train = dataset_train.shuffle(buffer_size=BUFFER_SIZE, seed=SEED)
dataset_train = dataset_train.repeat()
dataset_train = dataset_train.batch(BATCH_SIZE)
dataset_train = dataset_train.prefetch(buffer_size=AUTOTUNE)
print(dataset_train)
```

```
<PrefetchDataset shapes: ((None, 256, 256, 3), (None, 256, 256, 1)), types: (tf.float32, tf.float32)>
```

In [80]:
```
x = x_val['Image_Path']
y = x_val['Mask_Path']
# Validation Dataset
dataset_val = tf.data.Dataset.from_tensor_slices((x, y))
dataset_val = dataset_val.map(lambda x, y: preprocess_image_mask(x, y), num_parallel_calls=tf.data.exp
erimental.AUTOTUNE)
dataset_val = dataset_val.map(lambda x, y: load_image_val(x, y), num_parallel_calls=tf.data.experiment
al.AUTOTUNE)
dataset_val = dataset_val.shuffle(buffer_size=BUFFER_SIZE, seed=SEED)
dataset_val = dataset_val.repeat()
dataset_val = dataset_val.batch(BATCH_SIZE)
dataset_val = dataset_val.prefetch(buffer_size=AUTOTUNE)
print(dataset_val)
```

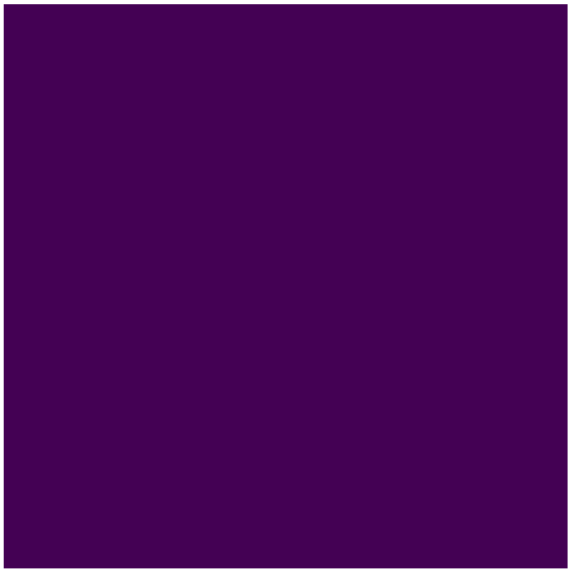
```
<PrefetchDataset shapes: ((None, 256, 256, 3), (None, 256, 256, 1)), types: (tf.float32, tf.float32)>
```

In [81]:
```
def display_sample(display_list):
    """Show side-by-side an input image,
    the ground truth and the prediction.
    """
    plt.figure(figsize=(18, 18))

    title = ['Input Image', 'True Mask', 'Predicted Mask']

    for i in range(len(display_list)):
        plt.subplot(1, len(display_list), i+1)
        plt.title(title[i])
        plt.imshow(tf.keras.preprocessing.image.array_to_img(display_list[i]))
        plt.axis('off')
    plt.show()
```

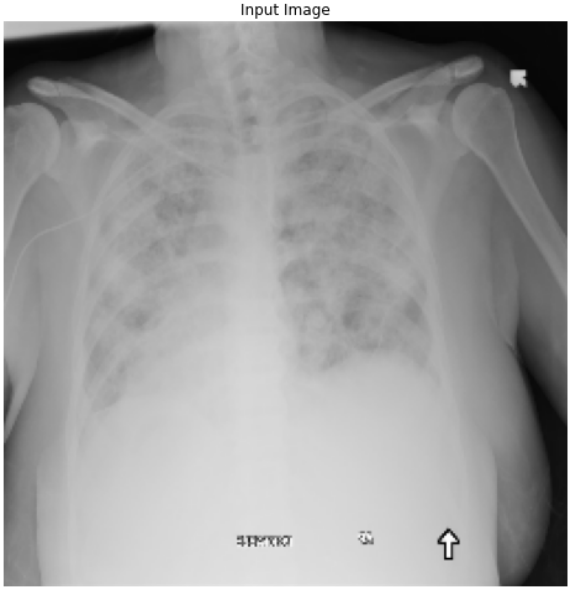
In [95]:
```
for image, mask in dataset_train.take(10):
  sample_image, sample_mask = image, mask
  display_sample([sample_image[0], sample_mask[0]])
```

Input Image

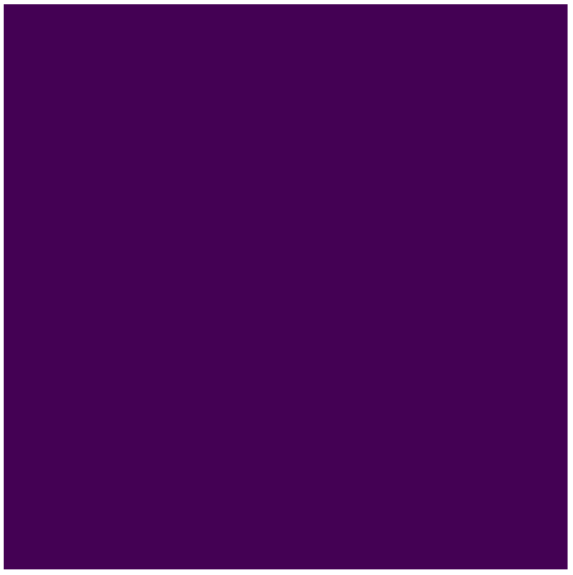
True Mask



Input Image

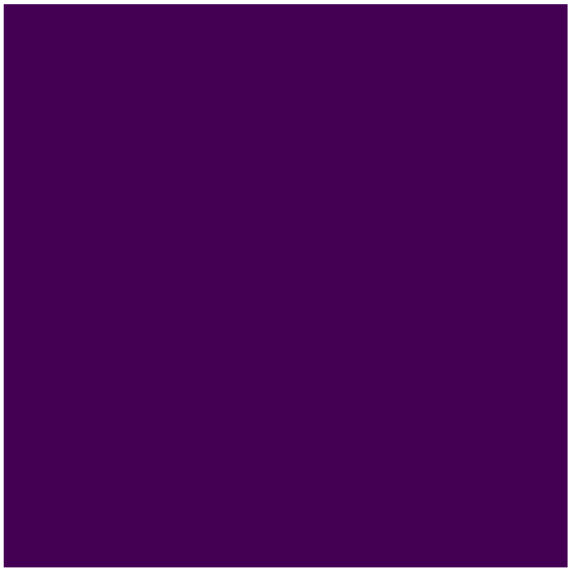


True Mask

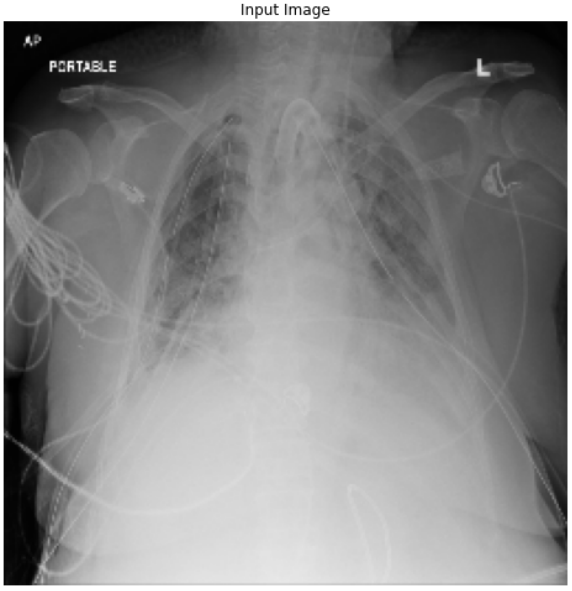


Input Image

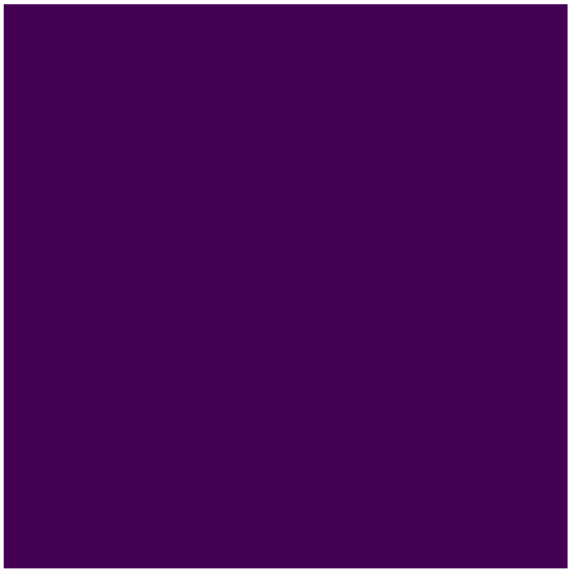
True Mask



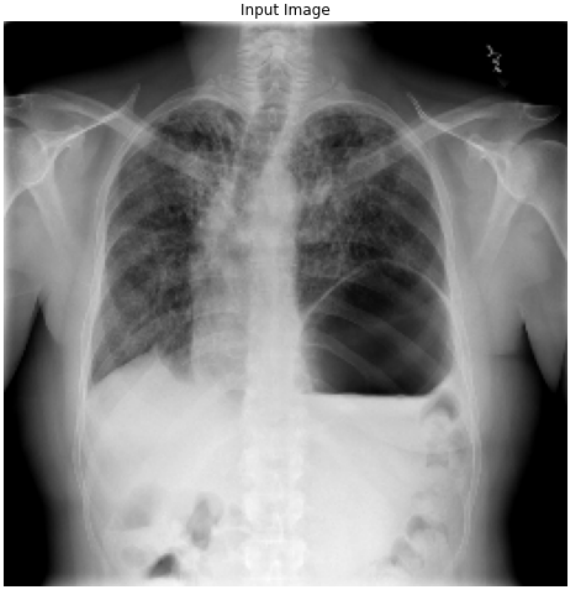
Input Image



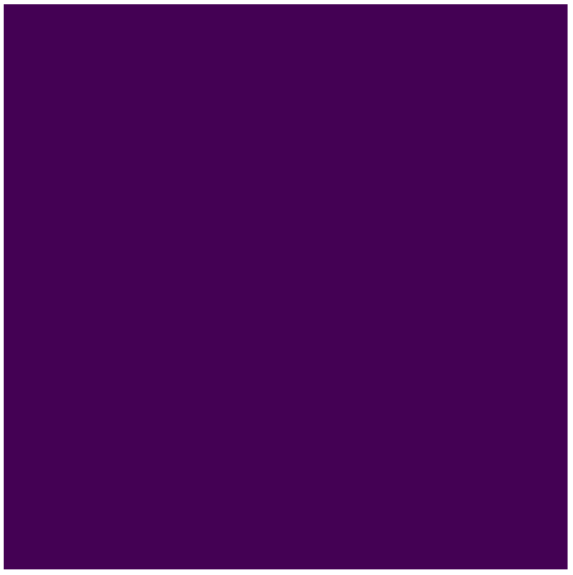
True Mask



Input Image

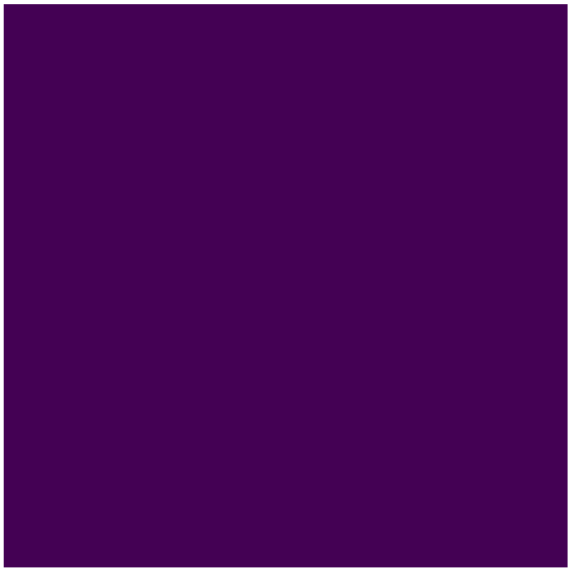


True Mask



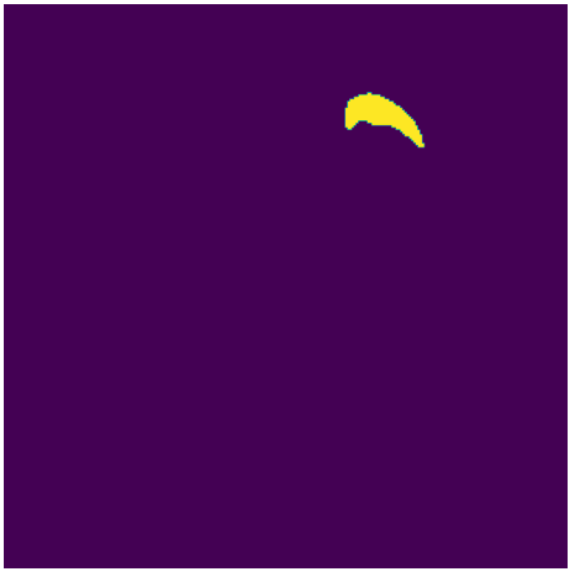
Input Image

True Mask

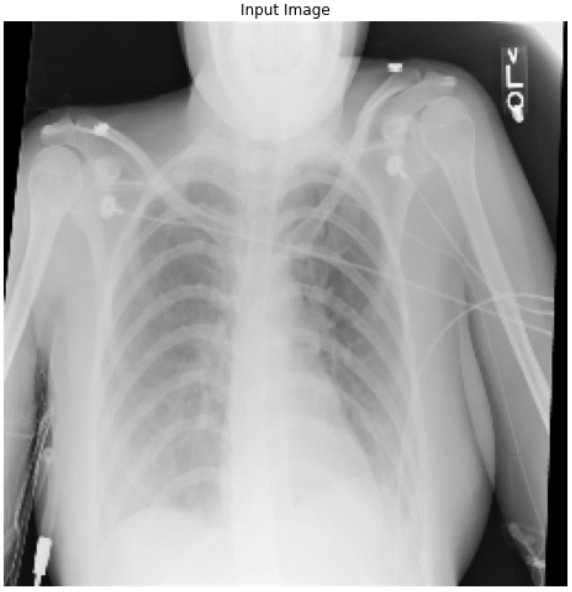


Input Image

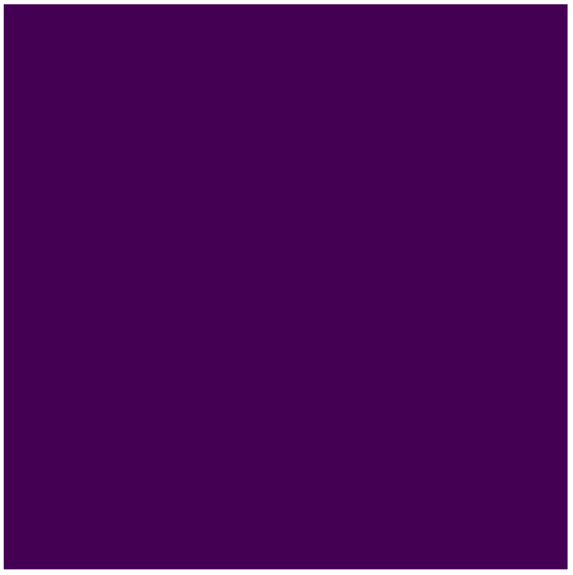
True Mask



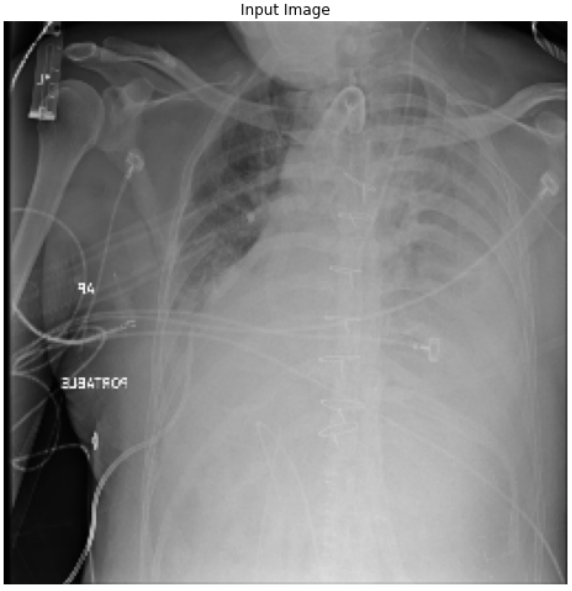
Input Image



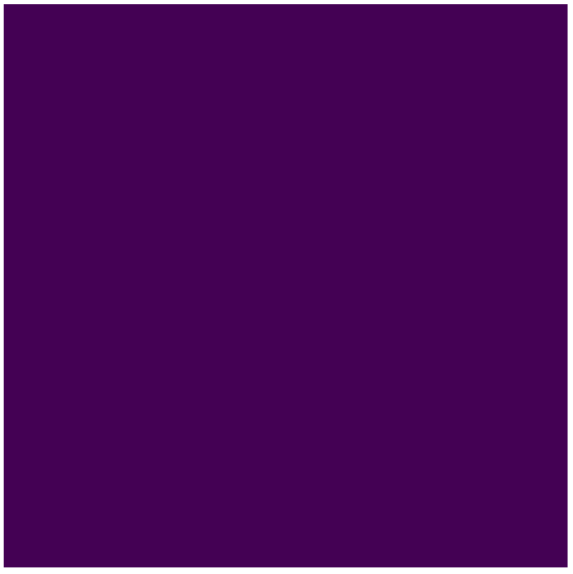
True Mask



Input Image



True Mask



Input Image

True Mask