

**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 warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import datetime as dt
from datetime import datetime
from tqdm.notebook import tqdm
from glob import glob
import pandas as pd
import shutil
import glob2
from tensorflow.keras import models, layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization, Activation, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import *
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
import datetime
from sklearn.model_selection import train_test_split
from keras.losses import binary_crossentropy
import keras.backend as K
from keras.models import load_model
```

```
In [2]: # install Libraries to read dicom images
!pip install -q tensorflow-io
!pip install pydicom
```

```
||██████████| 25.3MB 129kB/s
Collecting pydicom
  Downloading https://files.pythonhosted.org/packages/f4/15/df16546bc59bfca390cf072d473fb2c8acd423163
  6f64356593a63137e55/pydicom-2.1.2-py3-none-any.whl (1.9MB)
  ||██████████| 1.9MB 20.8MB/s
Installing collected packages: pydicom
Successfully installed pydicom-2.1.2
```

```
In [3]: import pydicom as dicom
import tensorflow as tf
import tensorflow_io as tfio
```

```
In [4]: # mount google drive
from google.colab import drive
drive.mount('gdrive', force_remount=True)
```

Mounted at gdrive

## Download the dataset from kaggle

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

```
In [5]: # download the dataset from kaggle
# https://www.kaggle.com/seesee/siim-train-test
!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/89.0.4389.90 Safari/537.36" --header="Accept: te
xt/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appli
cation/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=G00G4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%
2F20210324%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210324T022759Z&X-Goog-Expires=259199&X-Goog-
SignedHeaders=host&X-Goog-Signature=6537e07b49380396cf2a8773c646d3e4847a77f3f9e6d24612c369ee3962e3aaab
5e69f6e9ea89f09026dea49c0ea2818d9a29f5e713e0b25cba7445cbfe806668b81034ec3b93f88942ec5770e0e69c7c2387a4
fcc6ea770aa548f4e84d1e7f7d789e8581e5a78883165555fc729dbfeeeeca80c797157680c411dd8e045b95a5eb7b304d91f89
f4e56a9bc25d46f84a416d540b4ae097d7ac0512bcc6ca52495e135a86065aaec9e9fe7f0188a29d89f1c11775b84f8d64d8b
cb3a8641feb1f2e7473c02a91402da8df9784bd889855e0c274a65098a5abccb5cc0f926f02ed52330b438bc2a538c77d0fb9
492927c1ec7296b0f9828950b2ffe6f6a12e76" -c -O 'archive.zip'

--2021-03-24 15:19:50-- https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archiv
e.zip?X-Goog-Algorithm=G00G4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gservice
account.com%2F20210324%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210324T022759Z&X-Goog-Expires=2
59199&X-Goog-SignedHeaders=host&X-Goog-Signature=6537e07b49380396cf2a8773c646d3e4847a77f3f9e6d24612c3
69ee3962e3aaab5e69f6e9ea89f09026dea49c0ea2818d9a29f5e713e0b25cba7445cbfe806668b81034ec3b93f88942ec577
0e0e69c7c2387a4fcc6ea770aa548f4e84d1e7f7d789e8581e5a78883165555fc729dbfeeeeca80c797157680c411dd8e045b9
5a5eb7b304d91f89f4e56a9bc25d46f84a416d540b4ae097d7ac0512bcc6ca52495e135a86065aaec9e9fe7f0188a29d89f1
c11775b84f8d64d8bcb3a8641feb1f2e7473c02a91402da8df9784bd889855e0c274a65098a5abccb5cc0f926f02ed52330b
438bc2a538c77d0fb9492927c1ec7296b0f9828950b2ffe6f6a12e76
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.7.240, 172.217.8.16, 172.217.13.
80, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.7.240|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2059765561 (1.9G) [application/zip]
Saving to: 'archive.zip'

archive.zip      100%[=====] 1.92G 87.2MB/s    in 23s

2021-03-24 15:20:13 (86.0 MB/s) - 'archive.zip' saved [2059765561/2059765561]
```

```
In [6]: # unzip the dataset
!unzip -qq 'archive.zip'
```

```
In [7]: # read the given train csv file
image_df = pd.read_csv('siim/train-rle.csv')
image_df.head()
```

Out[7]:

	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...

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

```
In [9]: # create a directory for dicom images
images_dicom = 'siim/images_dicom/'
if not os.path.isdir(images_dicom):
    os.makedirs(images_dicom)

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

```
In [10]: # remove extra space in EncodedPixels column
image_df.rename(columns = {'EncodedPixels': 'EncodedPixels'}, inplace = True)

# add a column whether the image is with pneumothorax or without pneumothorax
image_df['is_pneumothorax'] = np.where(image_df['EncodedPixels'] == '-1', 0, 1)

image_df.head()
```

Out[10]:

	ImageId	EncodedPixels	is_pneumothorax
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	0
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	1
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	0
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	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...	1

```
In [11]: image_df = image_df.loc[image_df['is_pneumothorax'] == 1]
image_df.head()
```

Out[11]:

	ImageId	EncodedPixels	is_pneumothorax
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	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 ...	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...	1
10	1.2.276.0.7230010.3.1.4.8323329.3514.151787517...	759441 11 1010 15 1007 18 1005 19 1005 20 1003...	1
13	1.2.276.0.7230010.3.1.4.8323329.14008.15178752...	119368 98 923 102 908 118 903 126 896 133 889 ...	1

```
In [12]: # split the dataset and use val_df for final prediction
from sklearn.model_selection import train_test_split
train_df, val_df = train_test_split(image_df, test_size=0.2, random_state=42, shuffle=True)
```

```
In [13]: # add full dicom path to image_df
val_df['dicom_path'] = images_dicom + val_df['ImageId']+'.dcm'
val_df.head()
```

Out[13]:

	ImageId	EncodedPixels	is_pneumothorax	
1106	1.2.276.0.7230010.3.1.4.8323329.4703.151787518...	609496 23 991 48 971 83 936 103 918 111 910 11...	1	siim/images_dicom/1.2.27
7989	1.2.276.0.7230010.3.1.4.8323329.1033.151787516...	284490 38 976 58 956 73 941 87 877 149 845 182...	1	siim/images_dicom/1.2.27
8108	1.2.276.0.7230010.3.1.4.8323329.12294.15178752...	233827 23 994 37 981 47 971 57 960 66 955 70 9...	1	siim/images_dicom/1.2.27
11527	1.2.276.0.7230010.3.1.4.8323329.4628.151787518...	190212 2 1015 7 1009 13 992 12 1 18 980 19 100...	1	siim/images_dicom/1.2.27
3796	1.2.276.0.7230010.3.1.4.8323329.11498.15178752...	674944 3 1017 8 1014 10 1012 13 1008 16 1005 2...	1	siim/images_dicom/1.2.27

```
In [14]: # 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)
```

```
In [15]: # Create Directories for mask png files
mask_png = 'siim/mask_png/'

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

```
In [16]: import cv2
# define function to convert mask to png image
def 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)
masks_to_png(val_df[['ImageId', 'EncodedPixels']], mask_png)
```

```
In [17]: # add full png path to image_df
val_df['mask_path'] = mask_png + val_df['ImageId'] + '_mask.png'
val_df.head()
```

Out[17]:

	ImageId	EncodedPixels	is_pneumothorax	
1106	1.2.276.0.7230010.3.1.4.8323329.4703.151787518...	609496 23 991 48 971 83 936 103 918 111 910 11...	1	siim/images_dicom/1.2.27
7989	1.2.276.0.7230010.3.1.4.8323329.1033.151787516...	284490 38 976 58 956 73 941 87 877 149 845 182...	1	siim/images_dicom/1.2.27
8108	1.2.276.0.7230010.3.1.4.8323329.12294.15178752...	233827 23 994 37 981 47 971 57 960 66 955 70 9...	1	siim/images_dicom/1.2.27
11527	1.2.276.0.7230010.3.1.4.8323329.4628.151787518...	190212 2 1015 7 1009 13 992 12 1 18 980 19 100...	1	siim/images_dicom/1.2.27
3796	1.2.276.0.7230010.3.1.4.8323329.11498.15178752...	674944 3 1017 8 1014 10 1012 13 1008 16 1005 2...	1	siim/images_dicom/1.2.27

```
In [18]: def dice_loss(y_true, y_pred):
    smooth = 1.
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = y_true_f * y_pred_f
    score = (2. * K.sum(intersection) + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
    return 1. - score

def combined_bce_dice_loss(y_true, y_pred):
    return binary_crossentropy(y_true, y_pred) + dice_loss(y_true, y_pred)

def iou_score(y_true, y_pred):
    smooth = 1.
    def f(y_true, y_pred):
        intersection = (y_true * y_pred).sum()
        union = y_true.sum() + y_pred.sum() - intersection
        x = (intersection + smooth) / (union + smooth)
        x = x.astype(np.float32)
        return x
    return tf.numpy_function(f, [y_true, y_pred], tf.float32)
```

```
In [19]: # Load best weights to unet_imagenet_model
from keras.models import load_model
# custom_objects = optional dictionary mapping names (strings) to custom classes or functions to be considered during deserialization
unet_imagenet_model = load_model("gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/segmentation/weights-17-0.3066.hdf5",
                                  custom_objects={'combined_bce_dice_loss':combined_bce_dice_loss, "iou_score":iou_score})
```

```
In [22]: # define iou_score for a single image mask
def iou_score_single_image(pred_mask, mask):
    intersection = np.sum(np.logical_and(pred_mask, mask))
    union = np.sum(np.logical_or(pred_mask, mask))
    smooth = 1.
    iou = (intersection + smooth) / (union + smooth)
    iou = np.mean(iou)
    return iou
```

```
In [25]: val_image_path = val_df['dicom_path'].values
val_mask_path = val_df['mask_path'].values

iou_score_list = []
for row_no in tqdm(range(len(val_image_path))):
    img_path = val_image_path[row_no]
    msk_path = val_mask_path[row_no]

    size = 256
    image = tf.io.read_file(img_path)
    image = tfio.image.decode_dicom_image(image, dtype=tf.uint8,color_dim=True,scale='preserve')
    image = tf.image.convert_image_dtype(image, tf.float32)
    image = tf.squeeze(image,[0])
    image = tf.tile(image, tf.constant([1,1,3], tf.int32))
    image = tf.image.resize(image,size=[size,size])
    image = tf.expand_dims(image, axis=0)
    # print(image)

    mask = tf.io.read_file(msk_path)
    mask = tf.image.decode_png(mask, channels=1)
    mask = tf.image.resize(mask, [size, size])
    mask = tf.convert_image_dtype(mask, tf.float32)
    mask = tf.expand_dims(mask, axis=0)
    mask = mask[0]
    # print(mask)

    pred = unet_imagenet_model.predict(image)
    pred_mask = (pred[0]>0.5).astype(np.uint8)
    pred_mask = tf.convert_to_tensor(pred_mask)

    iou = iou_score_single_image(pred_mask, mask)
    iou_score_list.append(iou)
```

```
In [28]: # add iou score in validation dataframe
val_df['iou_score'] = iou_score_list
# sort val_df in descending order based on iou score
val_df = val_df.sort_values(by=['iou_score'], ascending=False)
val_df.head()
```

Out[28]:

		ImageId	EncodedPixels	is_pneumothorax	
10870	1.2.276.0.7230010.3.1.4.8323329.14439.15178752...		161066 32 977 52 962 63 951 74 945 80 939 86 9...	1	siim/images_dicom/1.2.27
7367	1.2.276.0.7230010.3.1.4.8323329.12911.15178752...		555163 8 1013 13 1008 18 1004 21 1001 24 998 2...	1	siim/images_dicom/1.2.27
223	1.2.276.0.7230010.3.1.4.8323329.698.1517875164...		257523 1 1019 4 1010 14 995 28 976 47 927 95 9...	1	siim/images_dicom/1.2.27
315	1.2.276.0.7230010.3.1.4.8323329.12438.15178752...		312629 41 976 58 961 70 951 73 949 76 947 77 9...	1	siim/images_dicom/1.2.27
8108	1.2.276.0.7230010.3.1.4.8323329.12294.15178752...		233827 23 994 37 981 47 971 57 960 66 955 70 9...	1	siim/images_dicom/1.2.27

```
In [39]: val_df.tail()
```

Out[39]:

		ImageId	EncodedPixels	is_pneumothorax	
573	1.2.276.0.7230010.3.1.4.8323329.11924.15178752...		579810 18 1001 28 992 36 985 43 979 47 975 52 ...	1	siim/images_dicom/1.2.276
3519	1.2.276.0.7230010.3.1.4.8323329.3317.151787517...		182990 2 1019 7 975 51 964 61 961 64 958 67 95...	1	siim/images_dicom/1.2.276
6553	1.2.276.0.7230010.3.1.4.8323329.10322.15178752...		587556 2 1012 12 1003 21 994 30 984 40 975 49 ...	1	siim/images_dicom/1.2.276
3949	1.2.276.0.7230010.3.1.4.8323329.10050.15178752...		489690 15 1004 20 1001 23 998 26 993 31 990 34...	1	siim/images_dicom/1.2.276
6997	1.2.276.0.7230010.3.1.4.8323329.11539.15178752...		254482 47 967 67 947 82 938 88 933 94 927 98 9...	1	siim/images_dicom/1.2.276

**Print 20 images with best IOU score**

```
In [36]: # print 20 images with best iou score from the top of the dataframe
for row_no in range(0,20):
    img_path = val_df['dicom_path'].iloc[row_no]
    msk_path = val_df['mask_path'].iloc[row_no]
    iou_sc = val_df['iou_score'].iloc[row_no]
    print("ImageID = " + str(img_path))
    print("IOU Score = " + str(iou_sc))

    size = 256
    image = tf.io.read_file(img_path)
    image = tfio.image.decode_dicom_image(image, dtype=tf.uint8,color_dim=True,scale='preserve')
    image = tf.image.convert_image_dtype(image, tf.float32)
    image = tf.squeeze(image,[0])
    image = tf.tile(image, tf.constant([1,1,3], tf.int32))
    image = tf.image.resize(image,size=[size,size])
    image = tf.expand_dims(image,axis=0)

    mask = tf.io.read_file(msk_path)
    mask = tf.image.decode_png(mask, channels=1)
    mask = tf.image.resize(mask, [size, size])
    mask = tf.image.convert_image_dtype(mask, tf.float32)
    mask = tf.expand_dims(mask, axis=0)

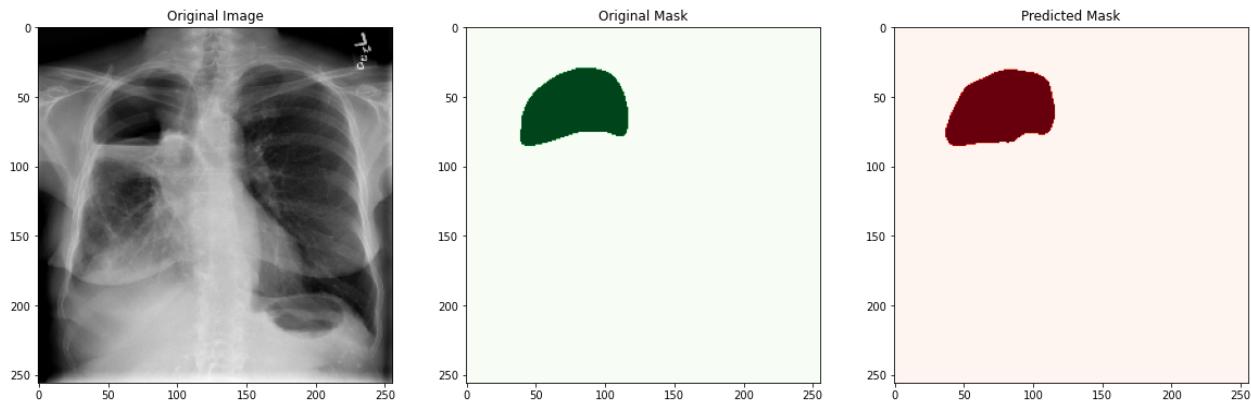
    # predict mask using unet_imagenet_model
    pred = unet_imagenet_model.predict(image)
    pred_mask = (pred[0]>0.5).astype(np.uint8)

    # plot original image
    plt.figure(figsize=(20,6))
    plt.subplot(131)
    plt.title("Original Image")
    plt.imshow(np.squeeze(image[0]),cmap='gray')

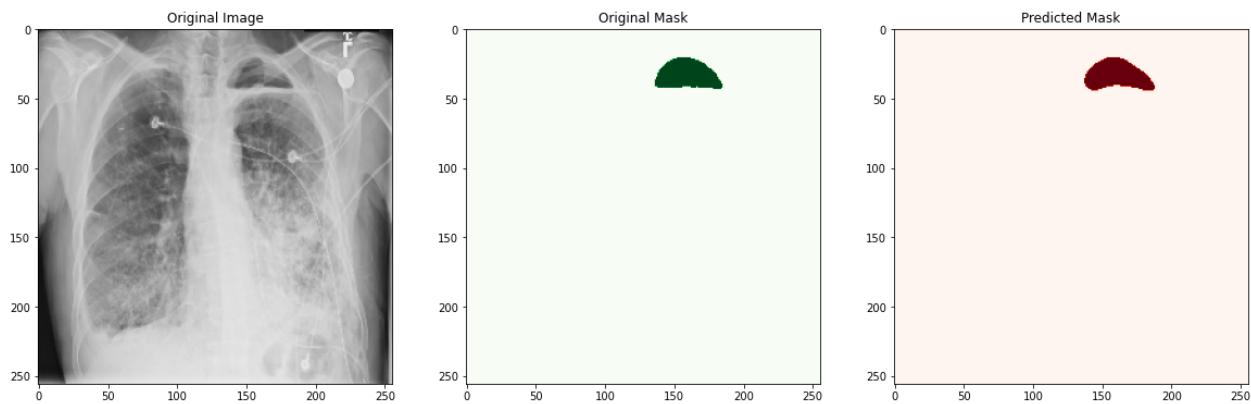
    # plot original mask
    plt.subplot(132)
    plt.title("Original Mask")
    plt.imshow(np.squeeze(mask[0]),cmap='Greens')

    # plot predicted mask
    plt.subplot(133)
    plt.title("Predicted Mask")
    plt.imshow(np.squeeze(pred_mask).astype(np.uint8),cmap='Reds')
    plt.show()
    print("\n\n")
```

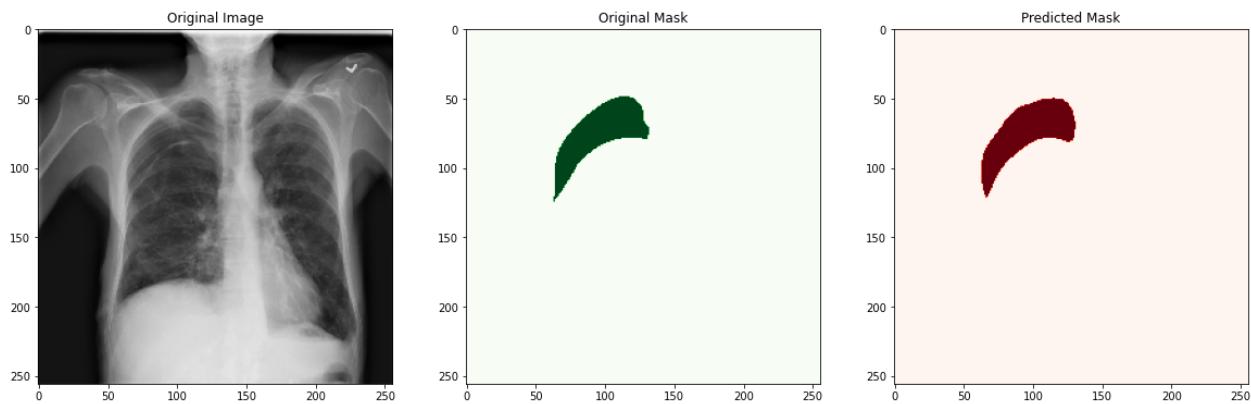
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.14439.1517875252.132263.dcm  
IOU Score = 0.9025386635541289



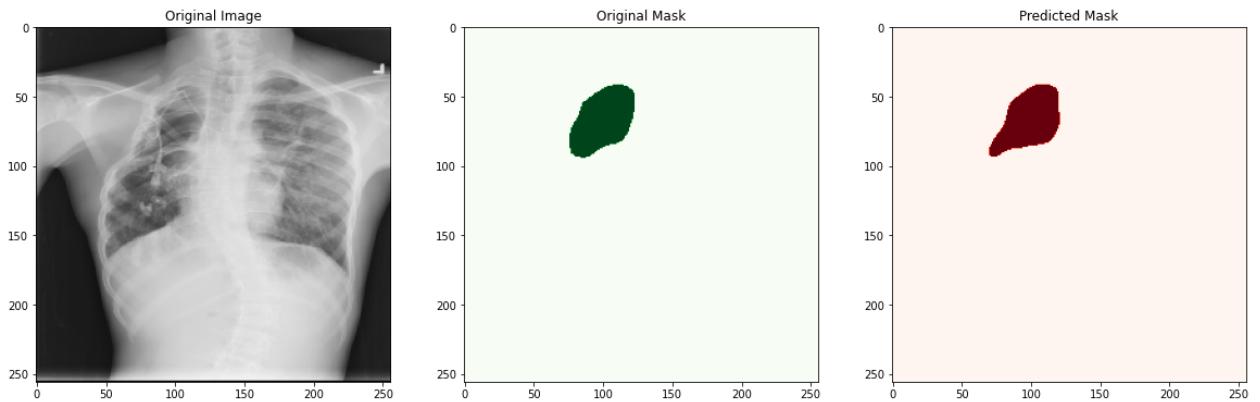
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12911.1517875242.492201.dcm  
IOU Score = 0.8878718535469108



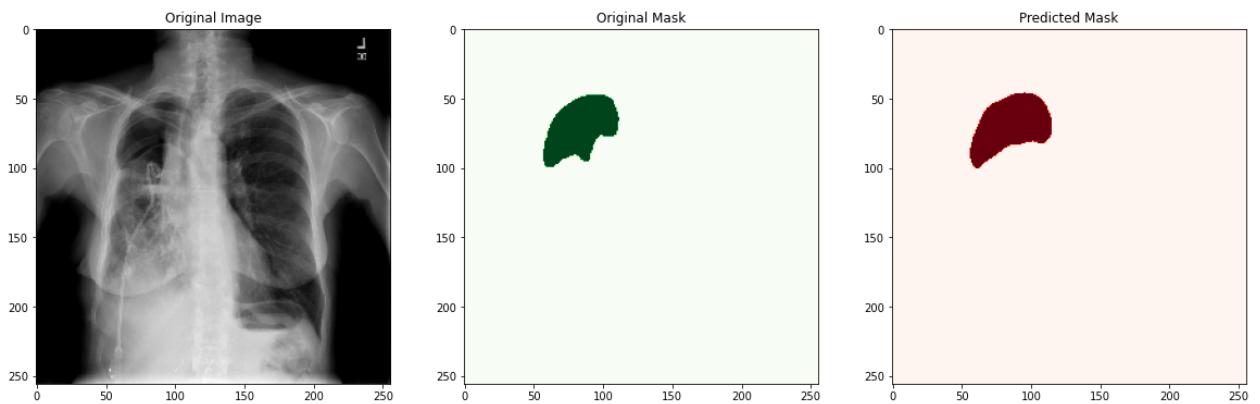
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.698.1517875164.123807.dcm  
IOU Score = 0.8856868395773295



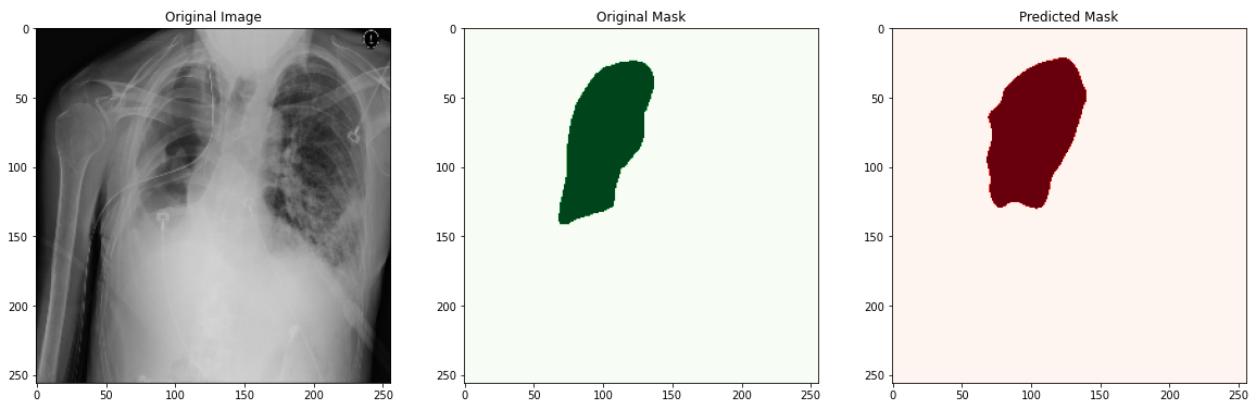
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12438.1517875239.123088.dcm  
IOU Score = 0.8543956043956044



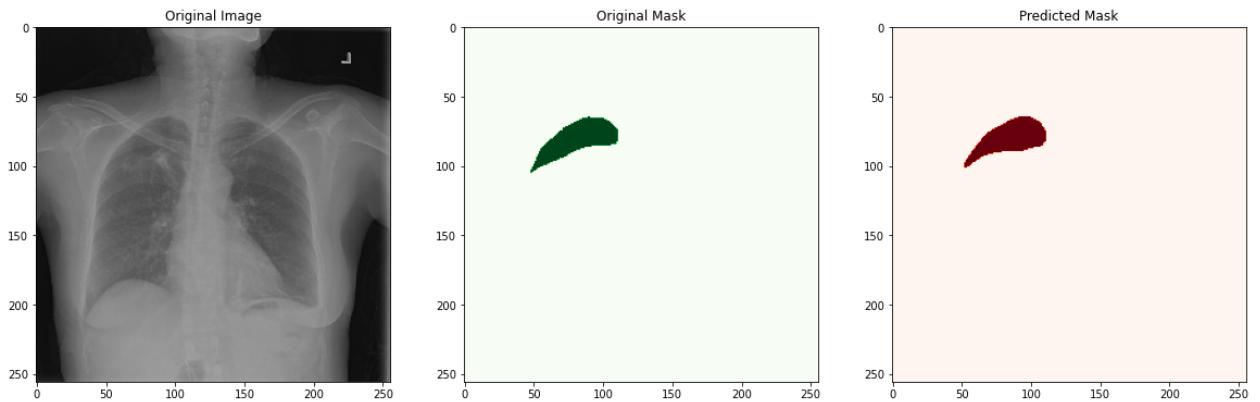
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12294.1517875238.418455.dcm  
IOU Score = 0.8186602870813398



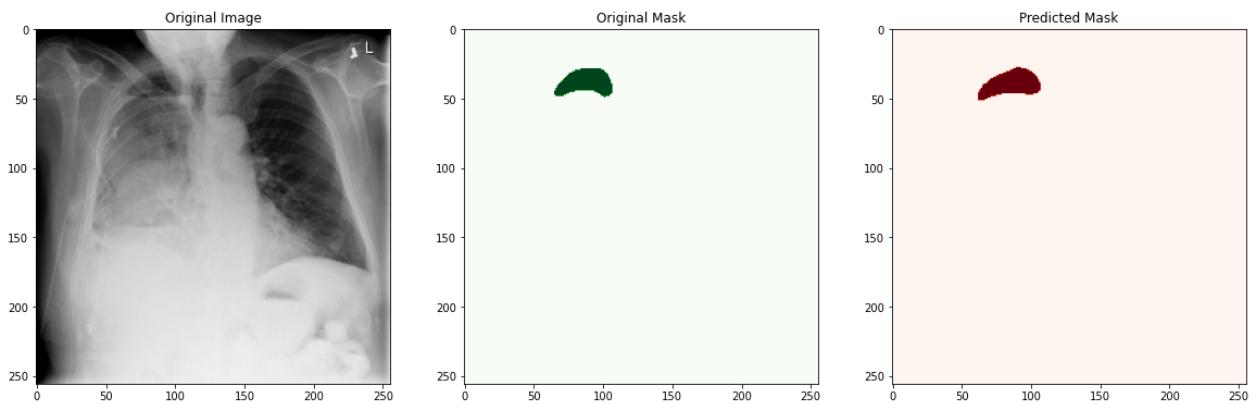
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.1337.1517875167.297636.dcm  
IOU Score = 0.8099016732792824



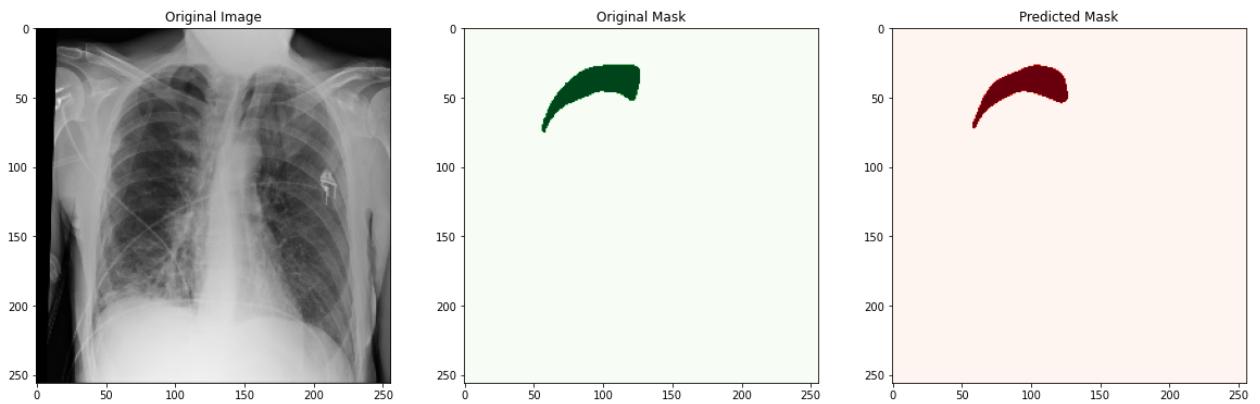
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.6141.1517875196.714183.dcm  
IOU Score = 0.8084745762711865



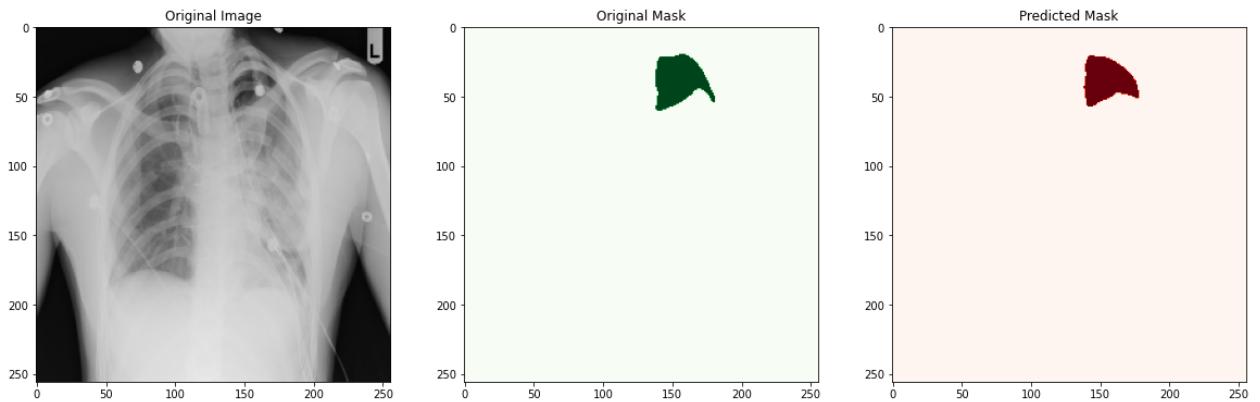
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4692.1517875184.149090.dcm  
IOU Score = 0.8037249283667621



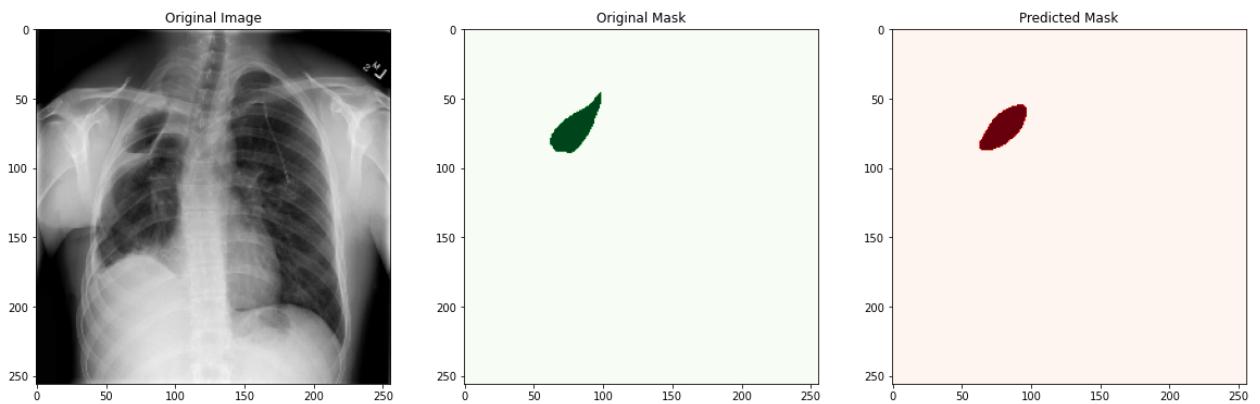
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.31720.1517875156.469363.dcm  
IOU Score = 0.7982520029133284



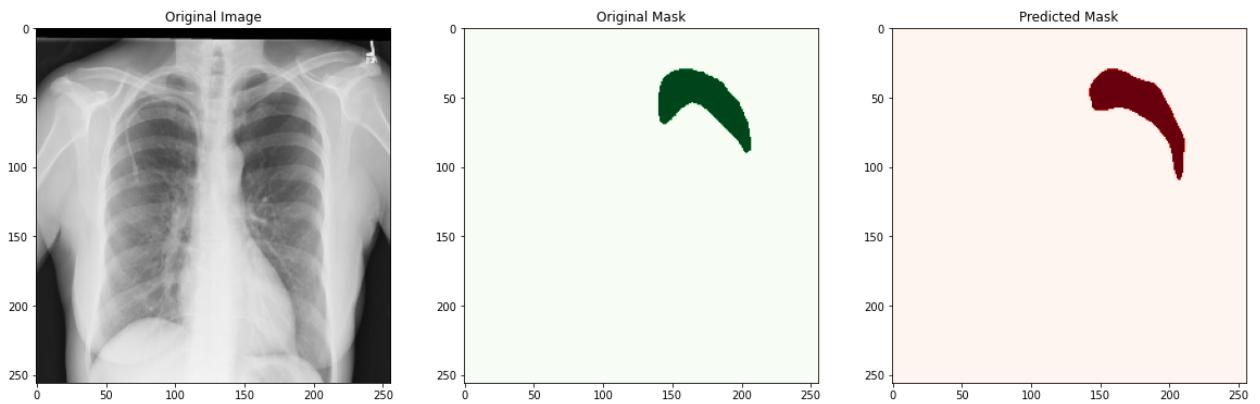
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.14040.1517875249.289553.dcm  
IOU Score = 0.788936170212766



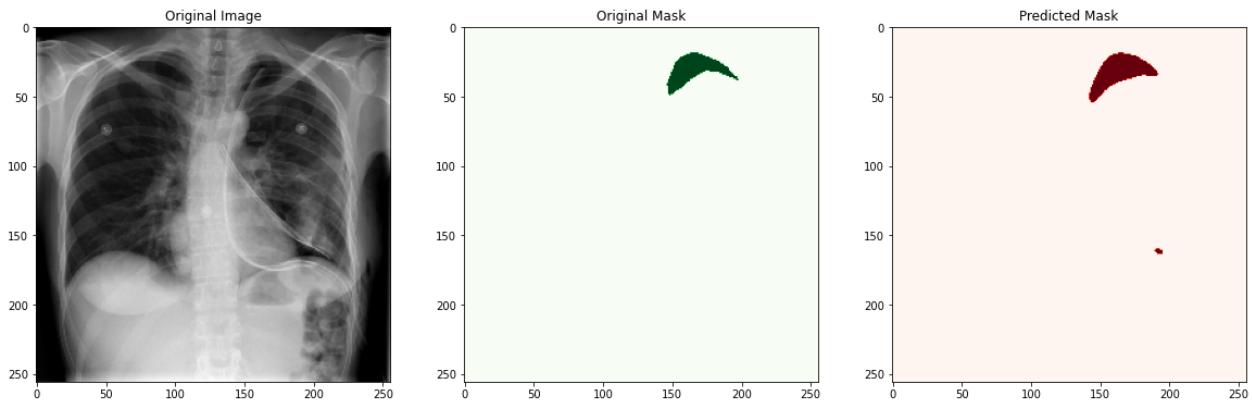
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3495.1517875178.82841.dcm  
 IOU Score = 0.7766497461928934



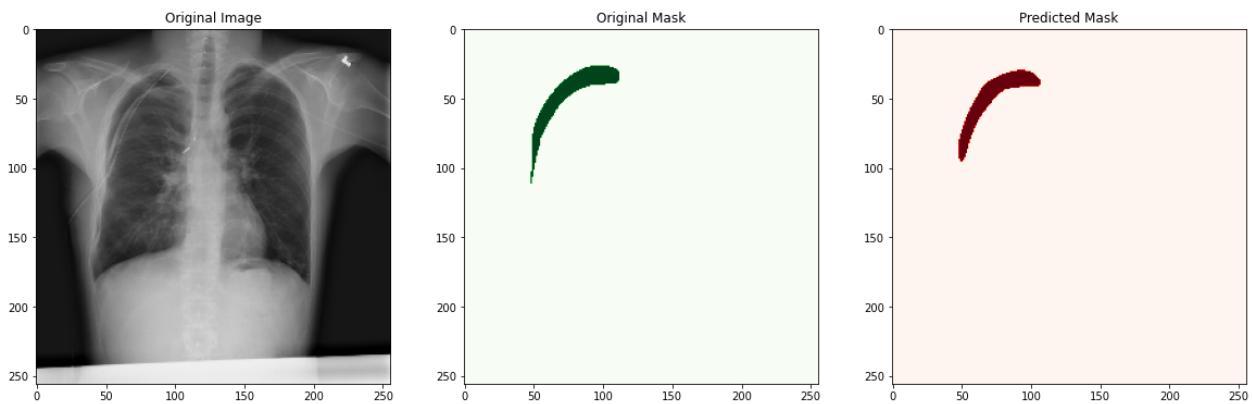
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.2466.1517875172.902109.dcm  
 IOU Score = 0.7524384579656294



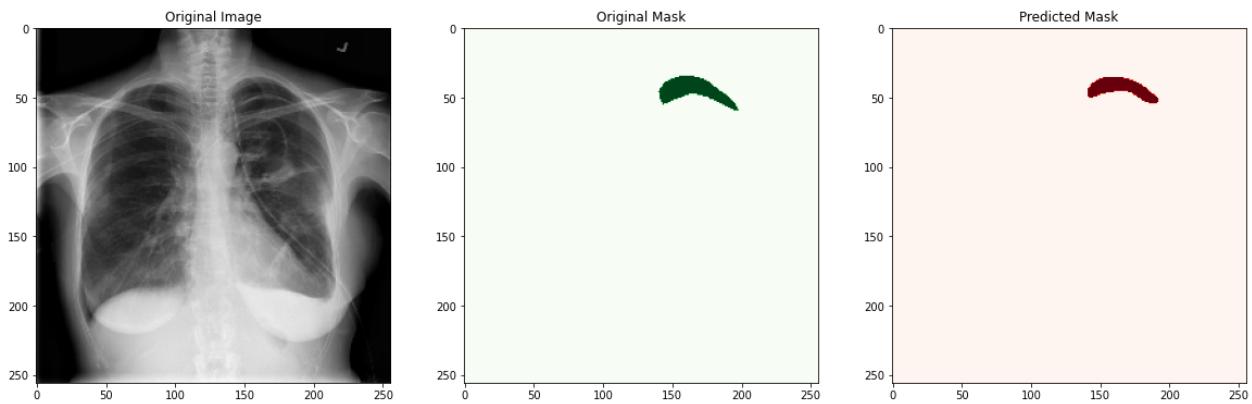
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.14235.1517875250.364895.dcm  
 IOU Score = 0.7416267942583732



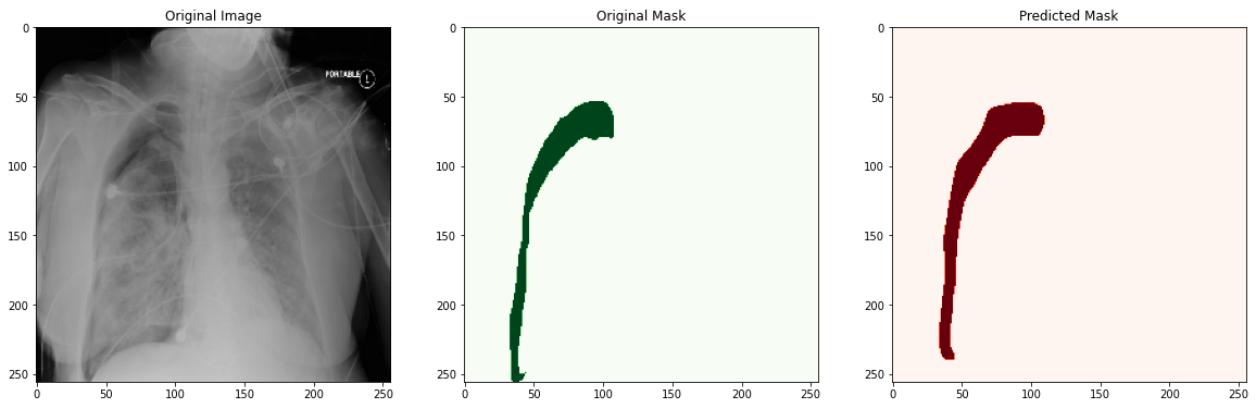
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.14158.1517875249.917785.dcm  
IOU Score = 0.7415215398716773



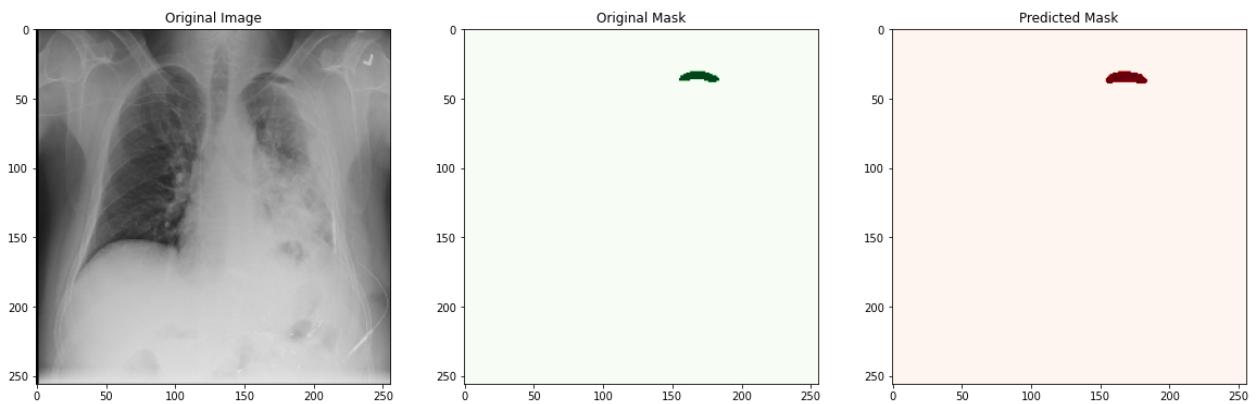
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.2665.1517875173.947393.dcm  
IOU Score = 0.7332339791356185



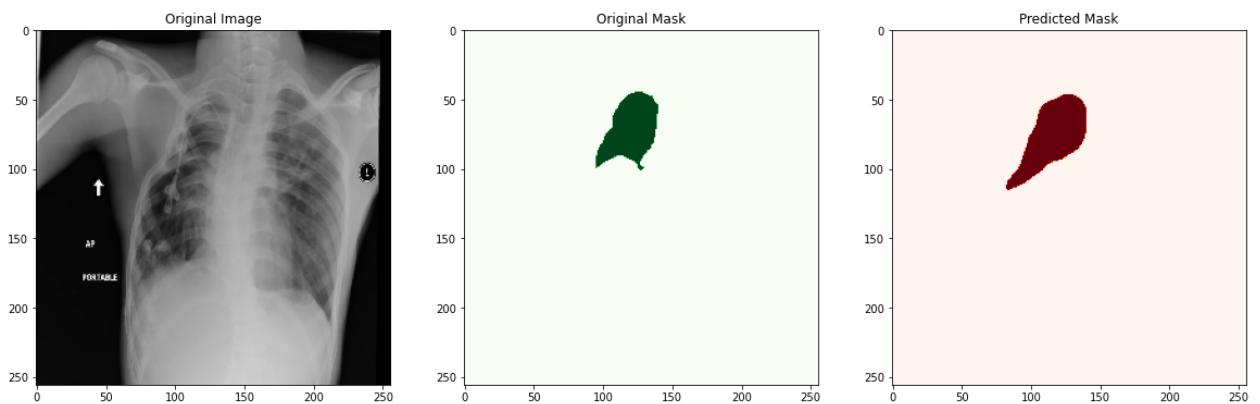
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.32518.1517875160.978553.dcm  
IOU Score = 0.7317073170731707



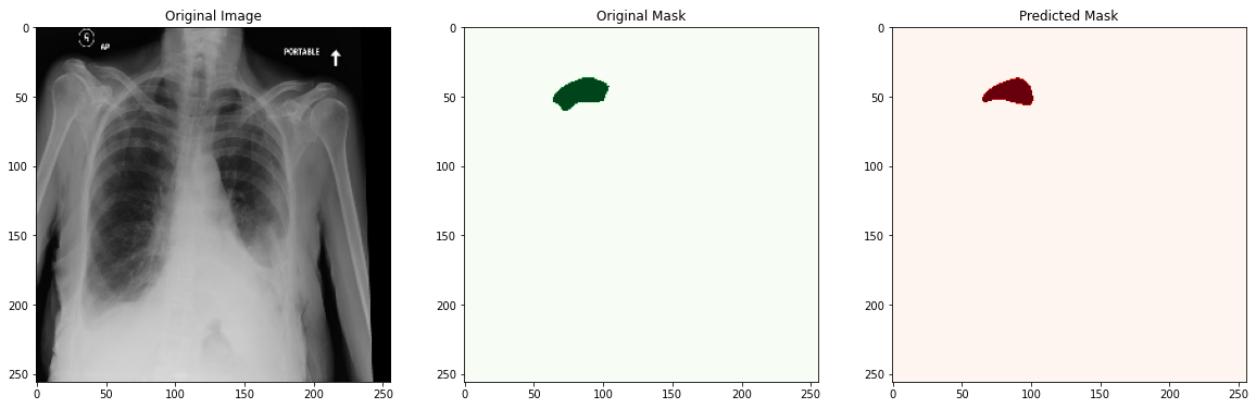
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3478.1517875178.16793.dcm  
IOU Score = 0.7307692307692307



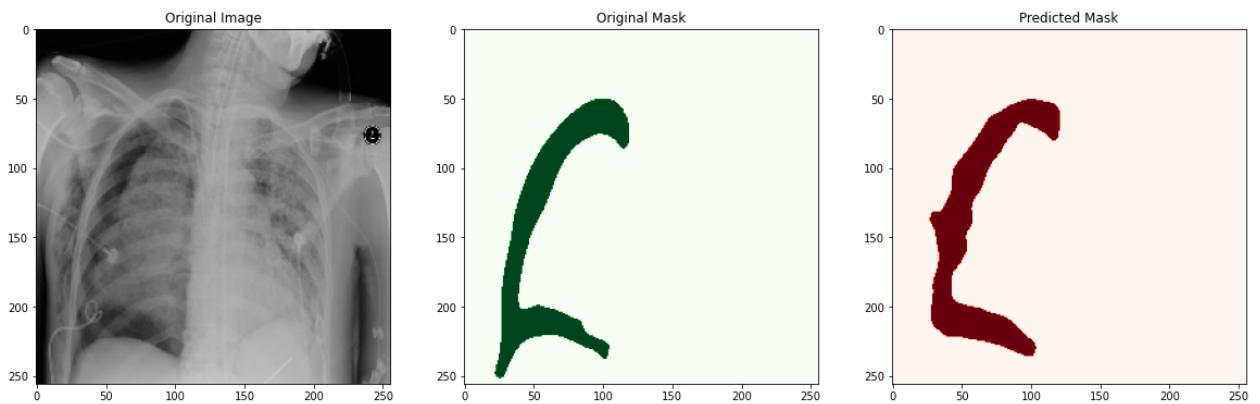
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4124.1517875181.222804.dcm  
IOU Score = 0.7260956175298805



ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10012.1517875220.965942.dcm  
IOU Score = 0.7232704402515723



ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.5527.1517875188.658681.dcm  
IOU Score = 0.712075344769593



**Print 20 images with worst IOU score**

```
In [37]: # print 20 images with worst iou score from the bottom of the dataframe
for row_no in range(len(val_df)-20,len(val_df)):
    img_path = val_df['dicom_path'].iloc[row_no]
    msk_path = val_df['mask_path'].iloc[row_no]
    iou_sc = val_df['iou_score'].iloc[row_no]
    print("ImageID = " + str(img_path))
    print("IOU Score = " + str(iou_sc))

    size = 256
    image = tf.io.read_file(img_path)
    image = tfio.image.decode_dicom_image(image, dtype=tf.uint8,color_dim=True,scale='preserve')
    image = tf.image.convert_image_dtype(image, tf.float32)
    image = tf.squeeze(image,[0])
    image = tf.tile(image, tf.constant([1,1,3], tf.int32))
    image = tf.image.resize(image,size=[size,size])
    image = tf.expand_dims(image,axis=0)

    mask = tf.io.read_file(msk_path)
    mask = tf.image.decode_png(mask, channels=1)
    mask = tf.image.resize(mask, [size, size])
    mask = tf.image.convert_image_dtype(mask, tf.float32)
    mask = tf.expand_dims(mask, axis=0)

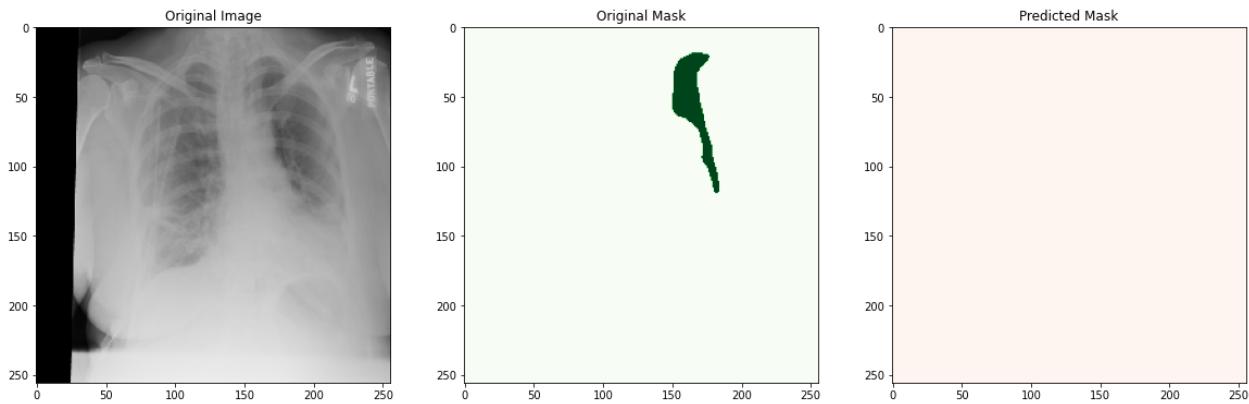
    # predict mask using unet_imagenet_model
    pred = unet_imagenet_model.predict(image)
    pred_mask = (pred[0]>0.5).astype(np.uint8)

    # plot original image
    plt.figure(figsize=(20,6))
    plt.subplot(131)
    plt.title("Original Image")
    plt.imshow(np.squeeze(image[0]),cmap='gray')

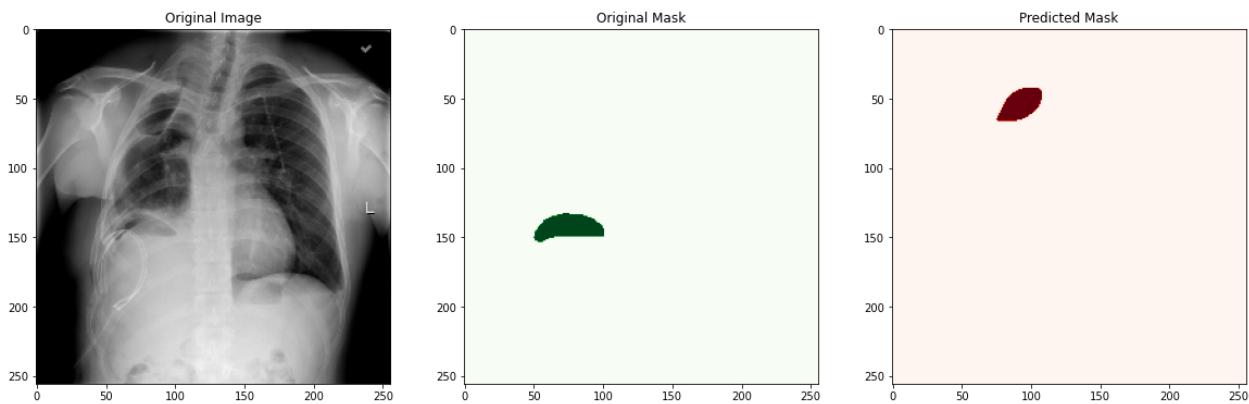
    # plot original mask
    plt.subplot(132)
    plt.title("Original Mask")
    plt.imshow(np.squeeze(mask[0]),cmap='Greens')

    # plot predicted mask
    plt.subplot(133)
    plt.title("Predicted Mask")
    plt.imshow(np.squeeze(pred_mask).astype(np.uint8),cmap='Reds')
    plt.show()
    print("\n\n")
```

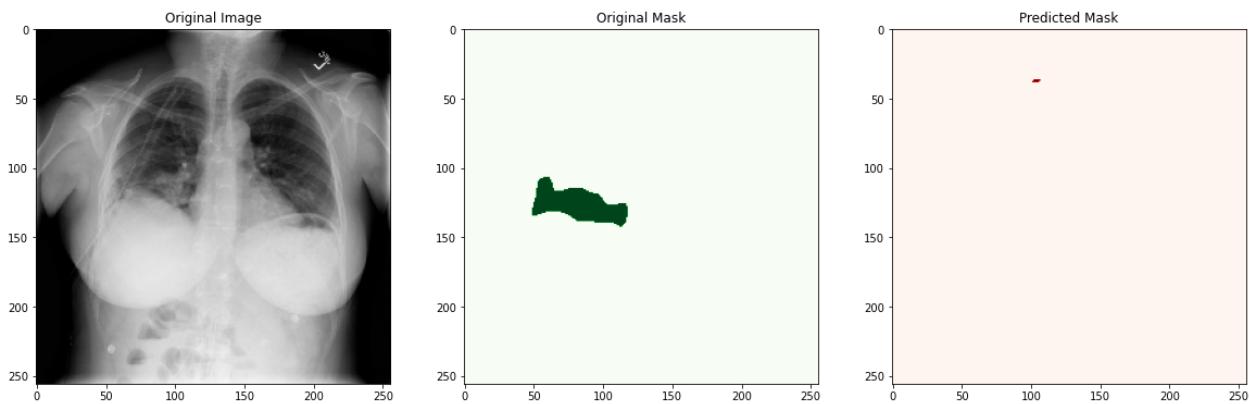
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.11756.1517875234.619205.dcm  
IOU Score = 0.0008084074373484236



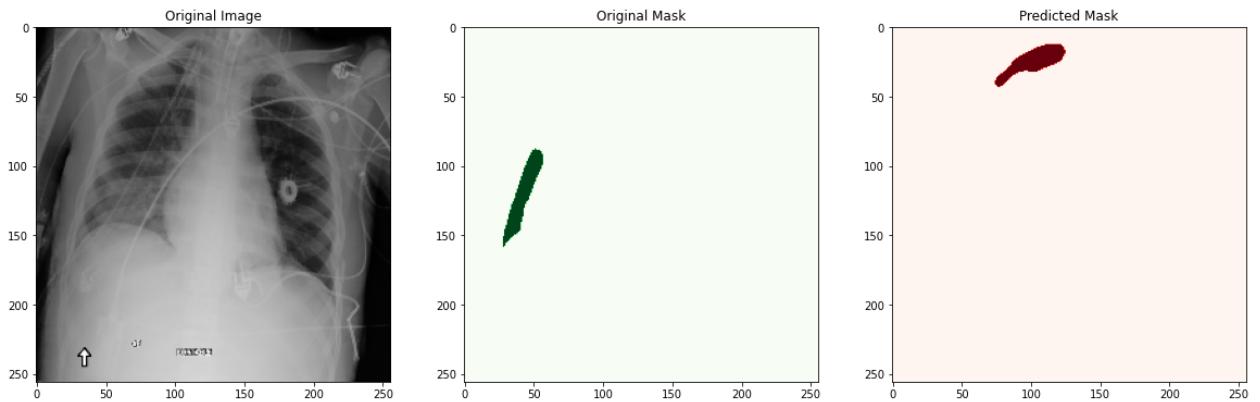
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.11599.1517875233.848287.dcm  
IOU Score = 0.0008032128514056225



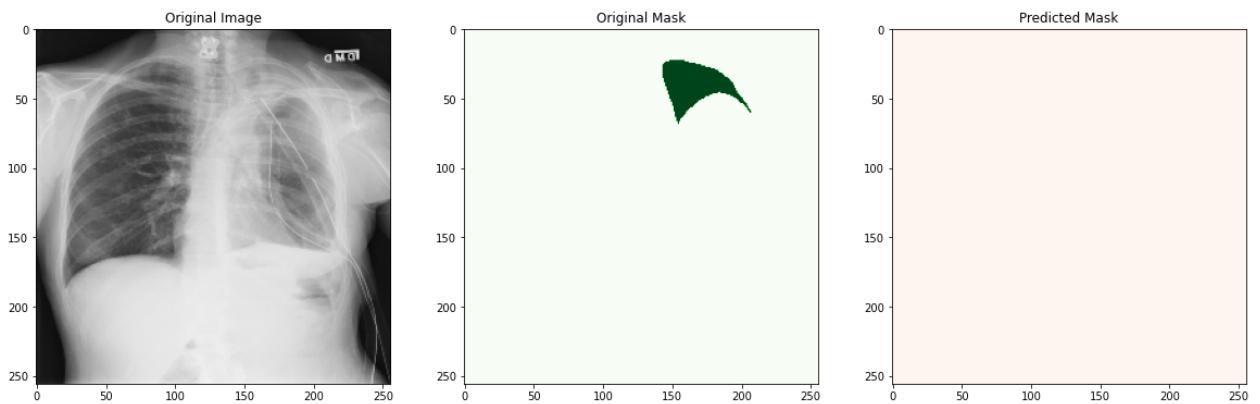
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4072.1517875180.971125.dcm  
IOU Score = 0.0007662835249042146



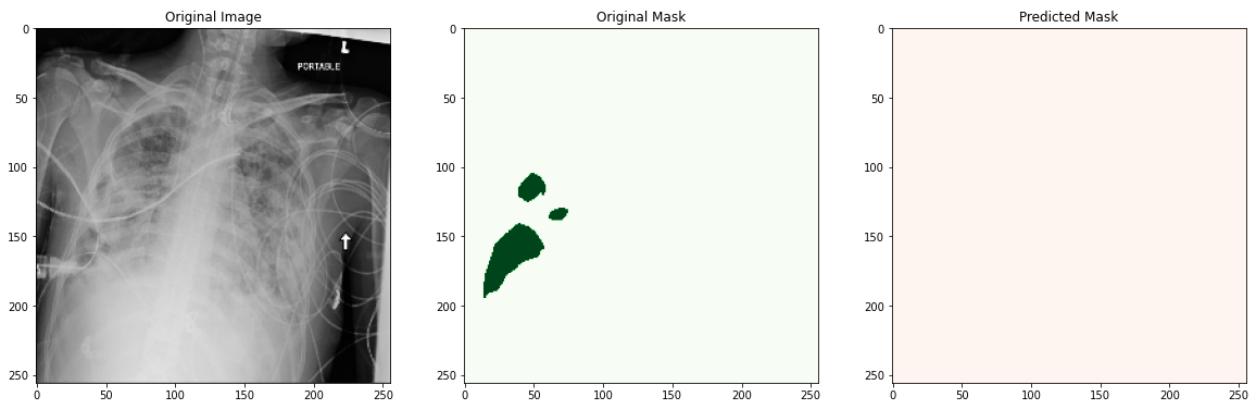
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3966.1517875180.399557.dcm  
IOU Score = 0.0007575757575757576



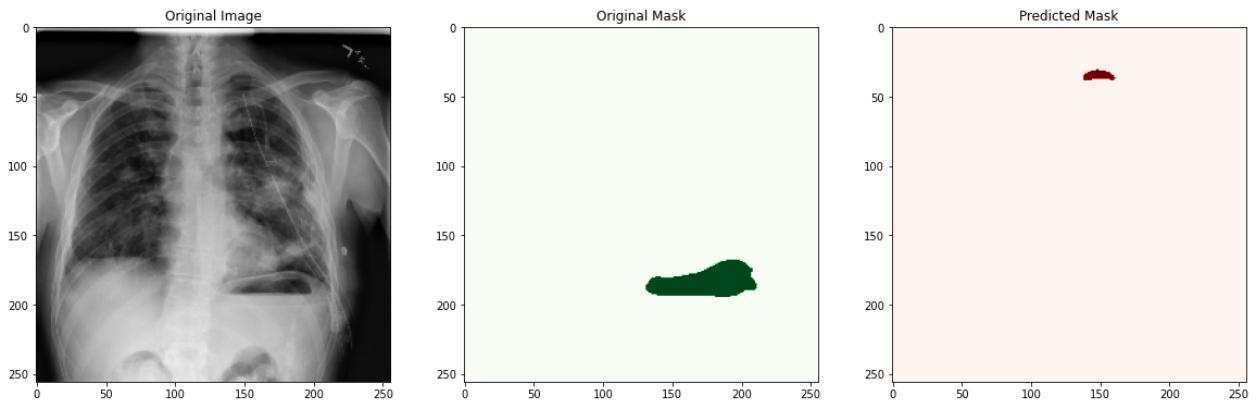
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.13676.1517875247.165344.dcm  
IOU Score = 0.0007530120481927711



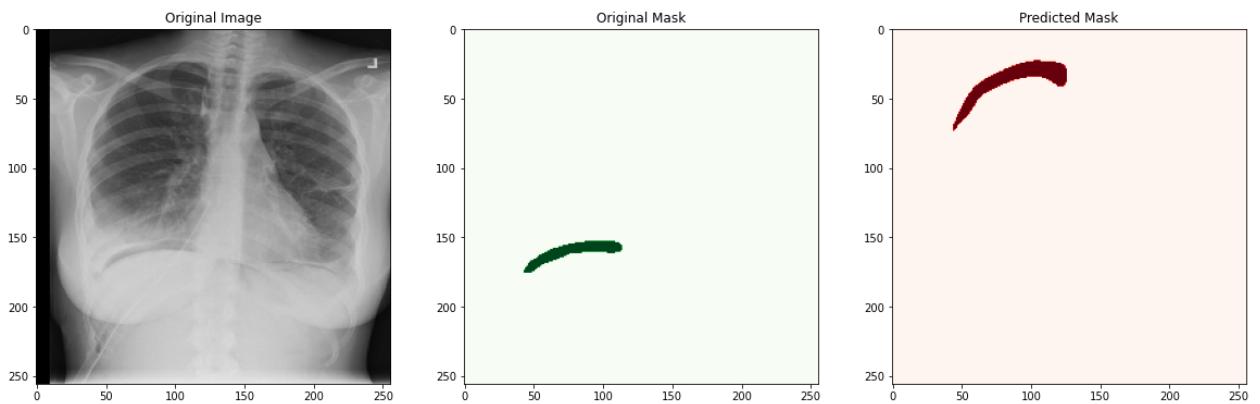
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.6490.1517875198.551179.dcm  
IOU Score = 0.0006693440428380187



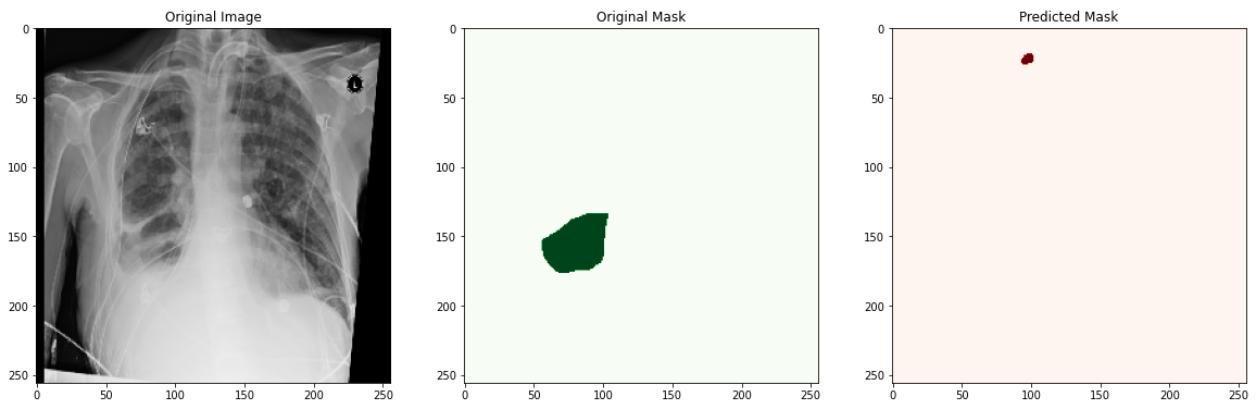
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.11607.1517875233.889082.dcm  
IOU Score = 0.0006583278472679394



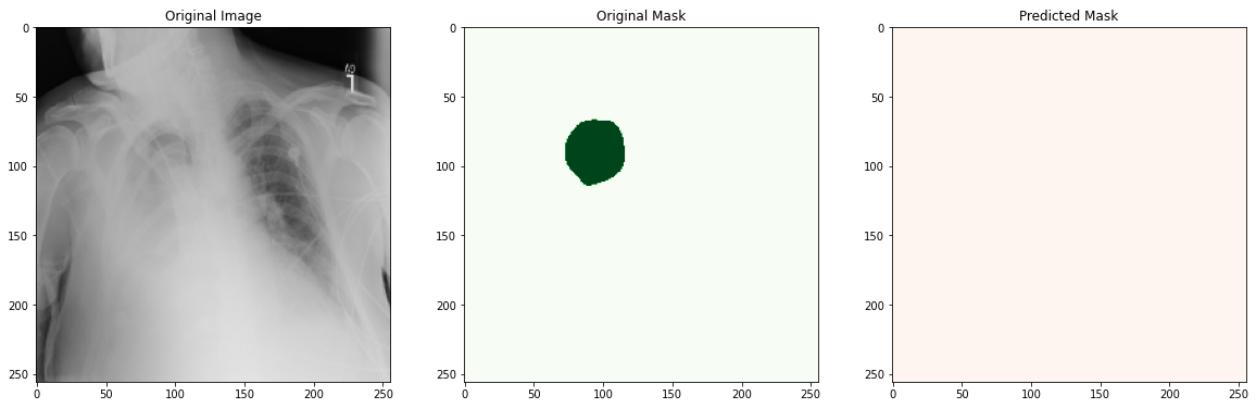
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4530.1517875183.330587.dcm  
IOU Score = 0.0006578947368421052



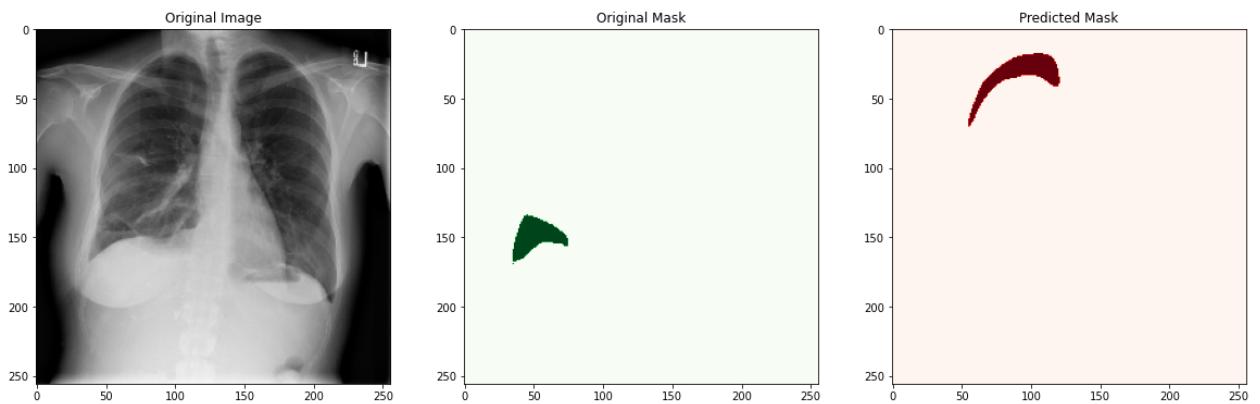
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.13557.1517875246.608609.dcm  
IOU Score = 0.0006381620931716656



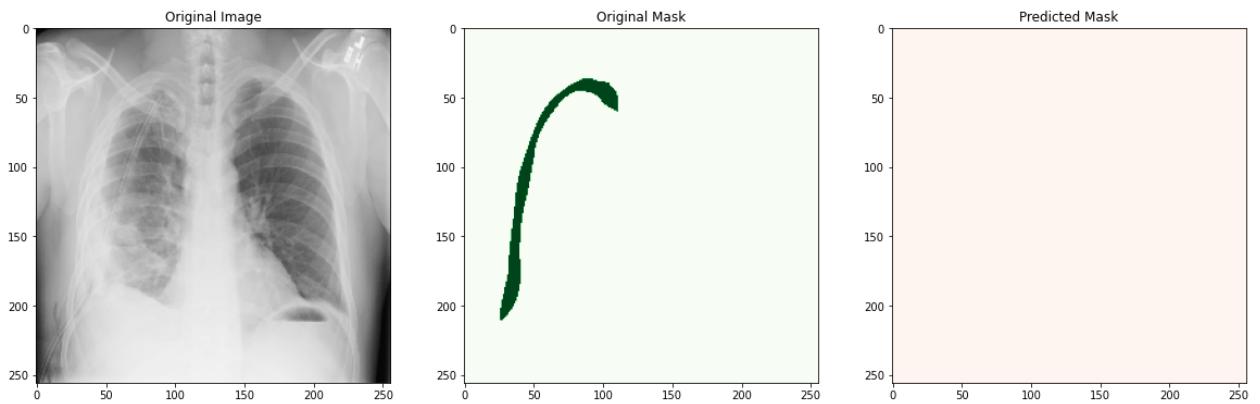
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.31892.1517875157.349249.dcm  
IOU Score = 0.0006176652254478073



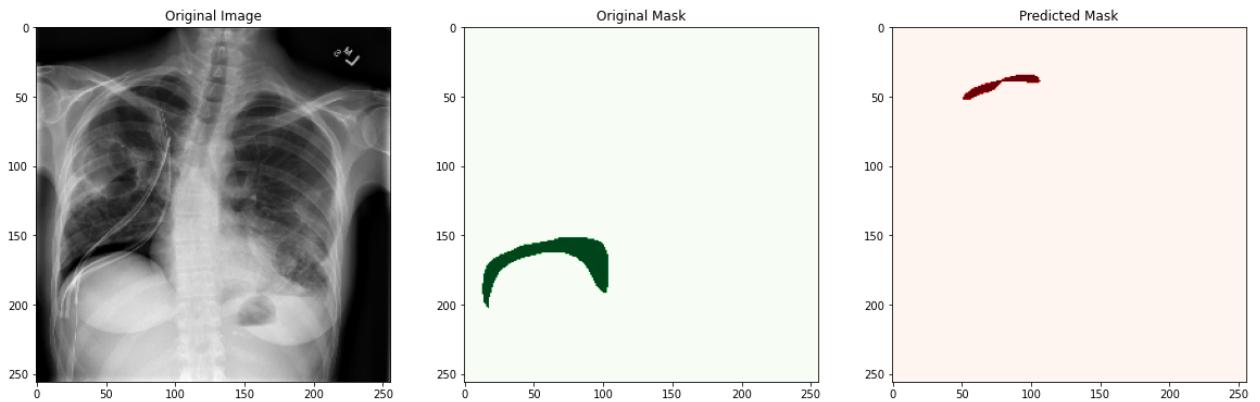
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10507.1517875224.16428.dcm  
 IOU Score = 0.0006105006105006105



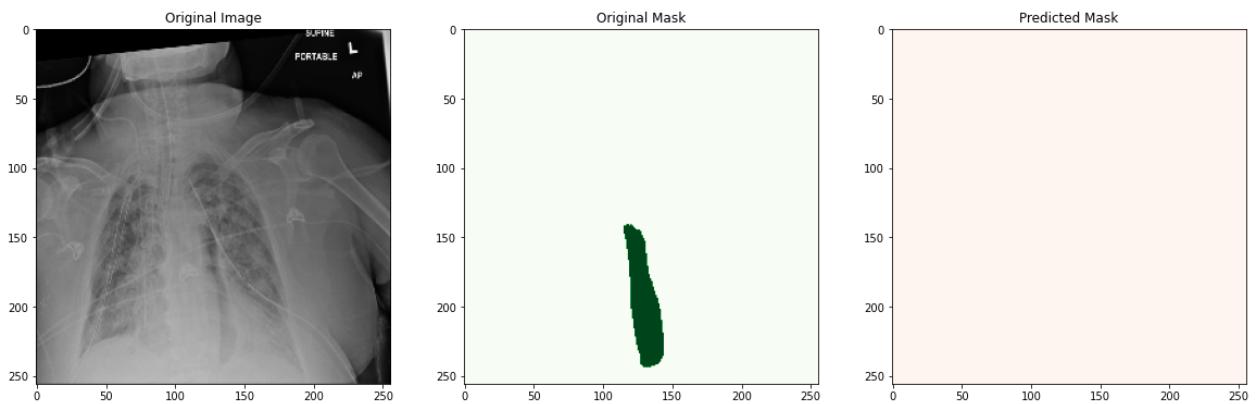
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12924.1517875242.544731.dcm  
 IOU Score = 0.0006024096385542169



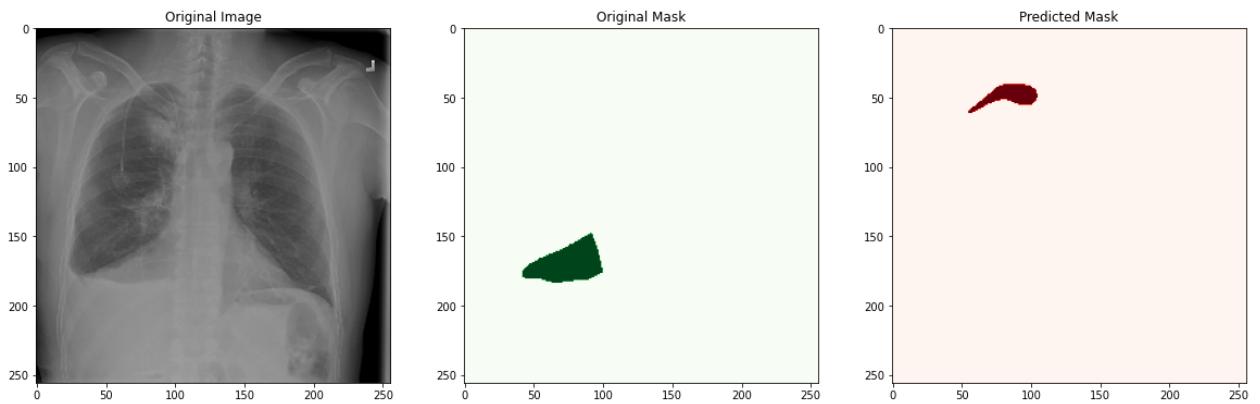
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.1320.1517875167.236100.dcm  
 IOU Score = 0.0005931198102016608



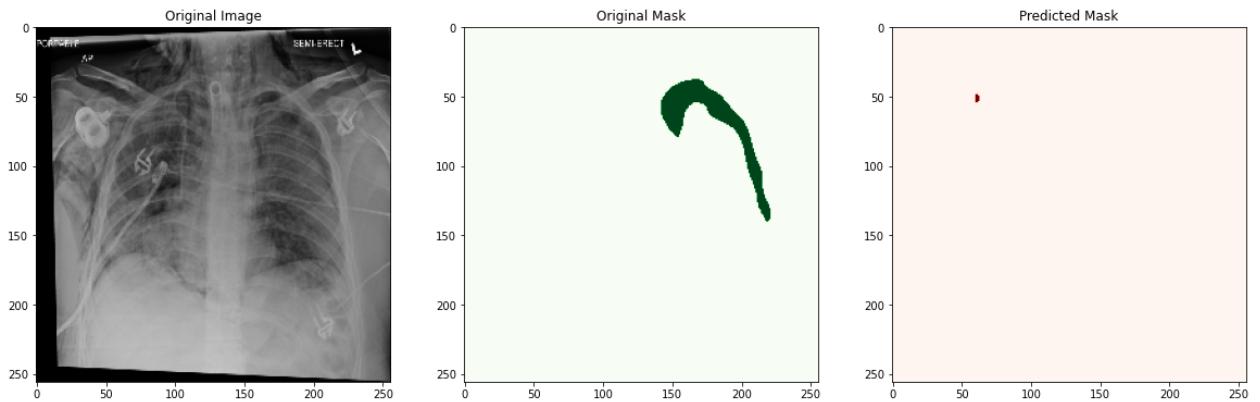
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.7010.1517875202.349776.dcm  
 IOU Score = 0.000591016548463357



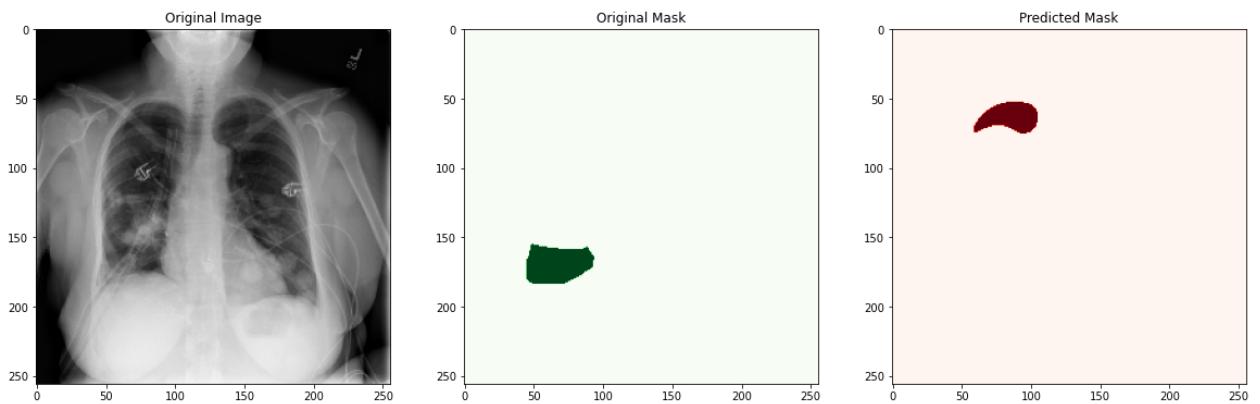
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12167.1517875237.806287.dcm  
 IOU Score = 0.0005899705014749262



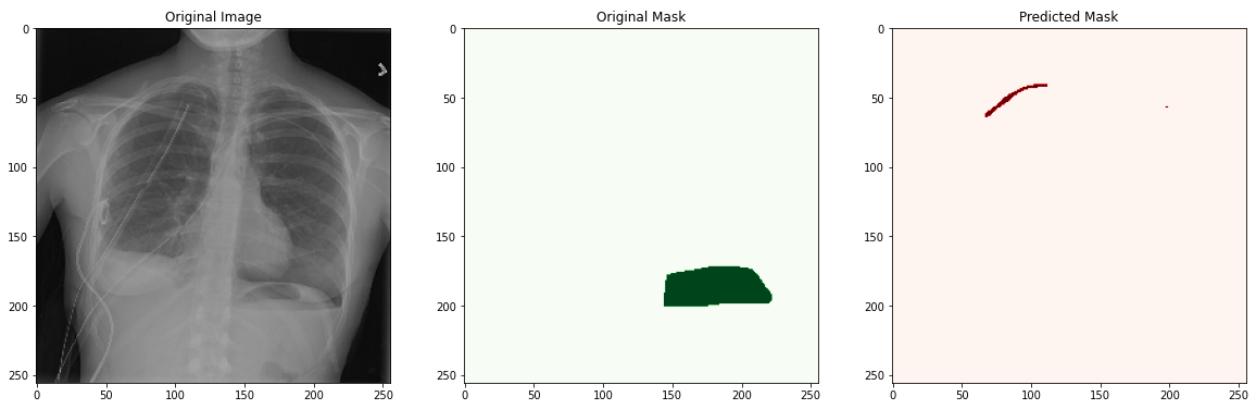
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.11924.1517875236.477550.dcm  
 IOU Score = 0.0005875440658049354



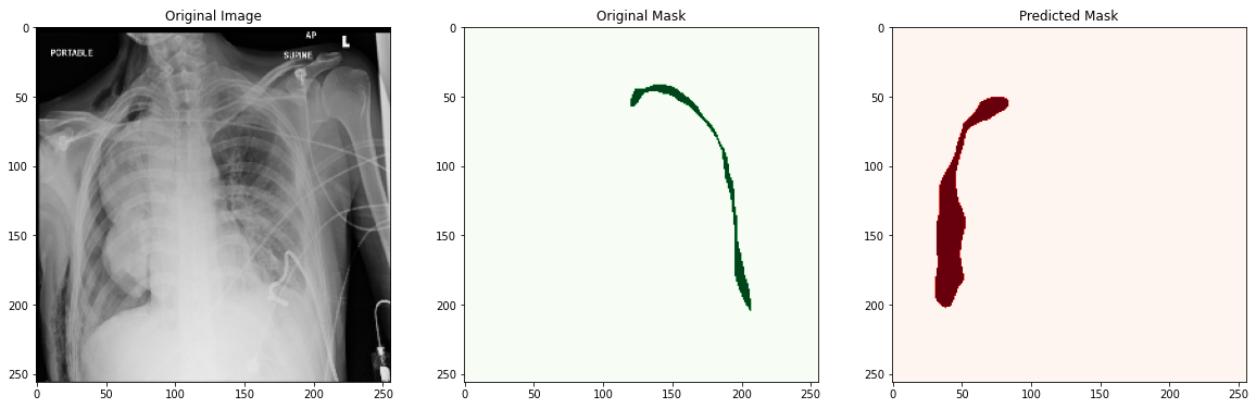
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3317.1517875177.225380.dcm  
IOU Score = 0.0005446623093681918



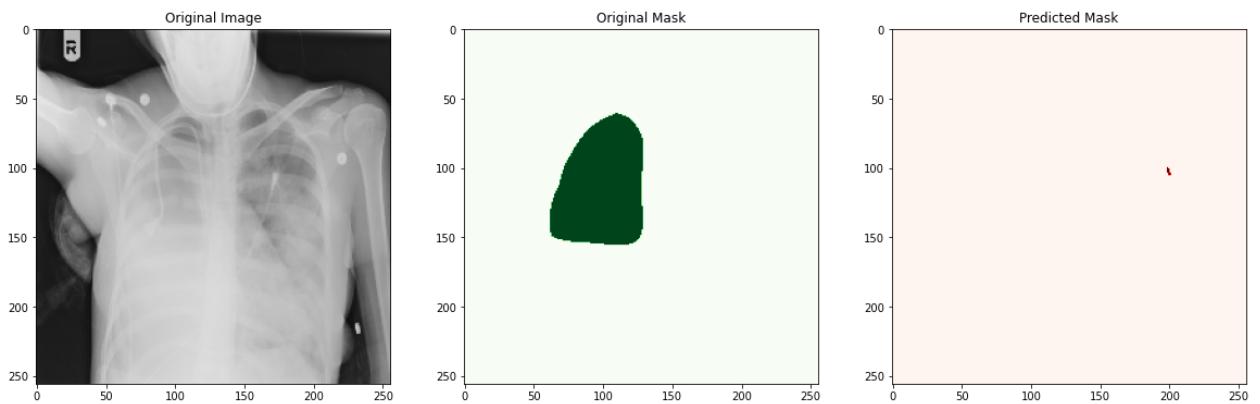
ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10322.1517875223.142884.dcm  
IOU Score = 0.0004940711462450593



ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10050.1517875221.763035.dcm  
IOU Score = 0.0003944331296673454

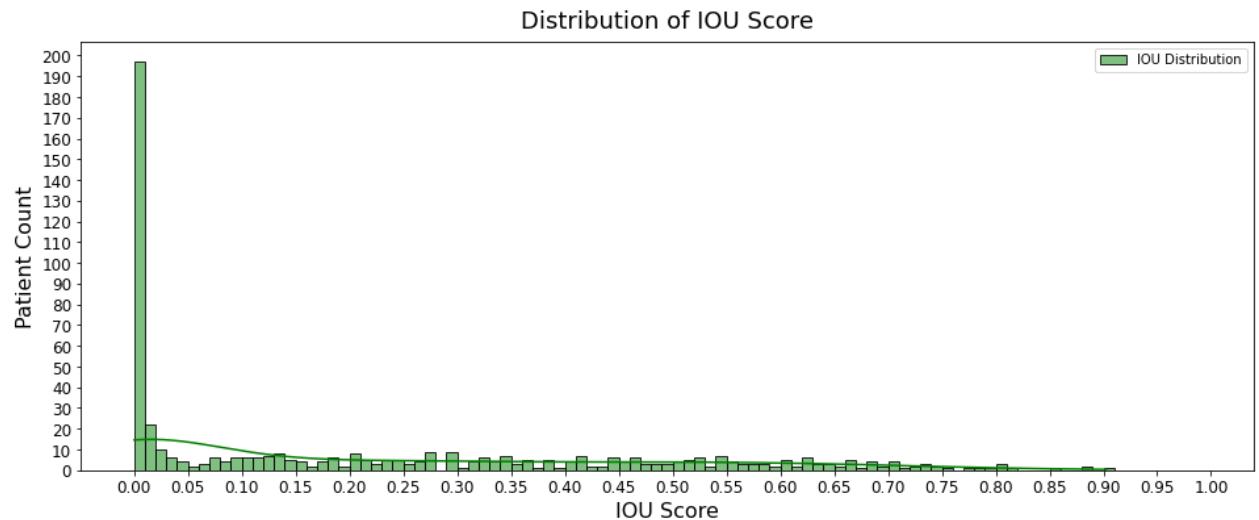


ImageID = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.11539.1517875233.482900.dcm  
IOU Score = 0.00019892580067634773



**Plot the distribution of the predicted probability score**

```
In [45]: # plot the distribution of the predicted probability score
plt.figure(figsize=(16, 6))
probability = [i/100 for i in range(100)]
sns.histplot(val_df['iou_score'], bins=[i for i in probability], color="green", kde=True, label="IOU Distribution")
plt.xticks([i/20 for i in range(21)], size = 12)
plt.yticks([i*10 for i in range(21)], size = 12)
plt.title("Distribution of IOU Score", fontsize=18, pad=10)
plt.xlabel("IOU Score", fontsize=16)
plt.ylabel("Patient Count", fontsize=16)
plt.legend()
plt.show()
```



## Observation:

1. There are around 200 images whose iou score is less than 0.1.
2. We need to train the model with more similar images for which iou score is very low, so that model can learn better.