```
#!pip install nibabel
#!pip install imageio
#!pip install ipywidgets
#!pip install fastai
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import glob
import nibabel as nib
import cv2
import imageio
from tqdm.notebook import tqdm
from ipywidgets import *
from PIL import Image
from fastai.basics import *
from fastai.vision.all import *
from fastai.data.transforms import *
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

```
# Create a meta file for nii files processing
file_list = []
for dirname, _, filenames in os.walk('/kaggle/input/liver-tumor-
segmentation'):
    for filename in filenames:
#print(os.path.join(dirname, filename))
        file_list.append((dirname,filename))
for dirname, _, filenames in os.walk('/kaggle/input/liver-tumor-
segmentation-part-2'):
    for filename in filenames:
        file_list.append((dirname,filename))
df_files = pd.DataFrame(file_list, columns =['dirname', 'filename'])
df_files.sort_values(by=['filename'], ascending=True)
```

```
                                                    dirname  \
89       /kaggle/input/liver-tumor-segmentation/segmentations
81       /kaggle/input/liver-tumor-segmentation/segmentations
142      /kaggle/input/liver-tumor-segmentation/segmentations
31       /kaggle/input/liver-tumor-segmentation/segmentations
45       /kaggle/input/liver-tumor-segmentation/segmentations
..                                                         ...
243  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
222  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
217  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
252  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
```

```
231  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6

                  filename
89     segmentation-0.nii
81     segmentation-1.nii
142   segmentation-10.nii
31   segmentation-100.nii
45   segmentation-101.nii
..                    ...
243         volume-95.nii
222         volume-96.nii
217         volume-97.nii
252         volume-98.nii
231         volume-99.nii

[262 rows x 2 columns]
```

```
# Map CT scan and label

df_files["mask_dirname"] = "" ; df_files["mask_filename"] = ""

for i in range(131):
    ct = f"volume-{i}.nii"
    mask = f"segmentation-{i}.nii"

    df_files.loc[df_files['filename'] == ct, 'mask_filename'] = mask
    df_files.loc[df_files['filename'] == ct, 'mask_dirname'] =
"../input/liver-tumor-segmentation/segmentations"

df_files_test= df_files[df_files.mask_filename=='']
# drop segment rows
df_files = df_files[df_files.mask_filename !=
''].sort_values(by=['filename']).reset_index(drop=True)
print(len(df_files))
df_files
#df_files_test
```

```
131

                                                  dirname
filename  \
0           /kaggle/input/liver-tumor-segmentation/volume_pt1
volume-0.nii
1           /kaggle/input/liver-tumor-segmentation/volume_pt1
volume-1.nii
2           /kaggle/input/liver-tumor-segmentation/volume_pt1
volume-10.nii
3    /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6  volume-
100.nii
4    /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
```

```
101.nii
..                                                   ...
...
126  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
volume-95.nii
127  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
volume-96.nii
128  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
volume-97.nii
129  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
volume-98.nii
130  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6
volume-99.nii

                                        mask_dirname
mask_filename
0    ../input/liver-tumor-segmentation/segmentations    segmentation-
0.nii
1    ../input/liver-tumor-segmentation/segmentations    segmentation-
1.nii
2    ../input/liver-tumor-segmentation/segmentations   segmentation-
10.nii
3    ../input/liver-tumor-segmentation/segmentations  segmentation-
100.nii
4    ../input/liver-tumor-segmentation/segmentations  segmentation-
101.nii
..                                                 ...
...
126  ../input/liver-tumor-segmentation/segmentations   segmentation-
95.nii
127  ../input/liver-tumor-segmentation/segmentations   segmentation-
96.nii
128  ../input/liver-tumor-segmentation/segmentations   segmentation-
97.nii
129  ../input/liver-tumor-segmentation/segmentations   segmentation-
98.nii
130  ../input/liver-tumor-segmentation/segmentations   segmentation-
99.nii

[131 rows x 4 columns]
```

```
def read_nii(filepath):
    '''
    Reads .nii file and returns pixel array
    '''
    ct_scan = nib.load(filepath)
    array   = ct_scan.get_fdata()
    array   = np.rot90(np.array(array))
    return(array)
```

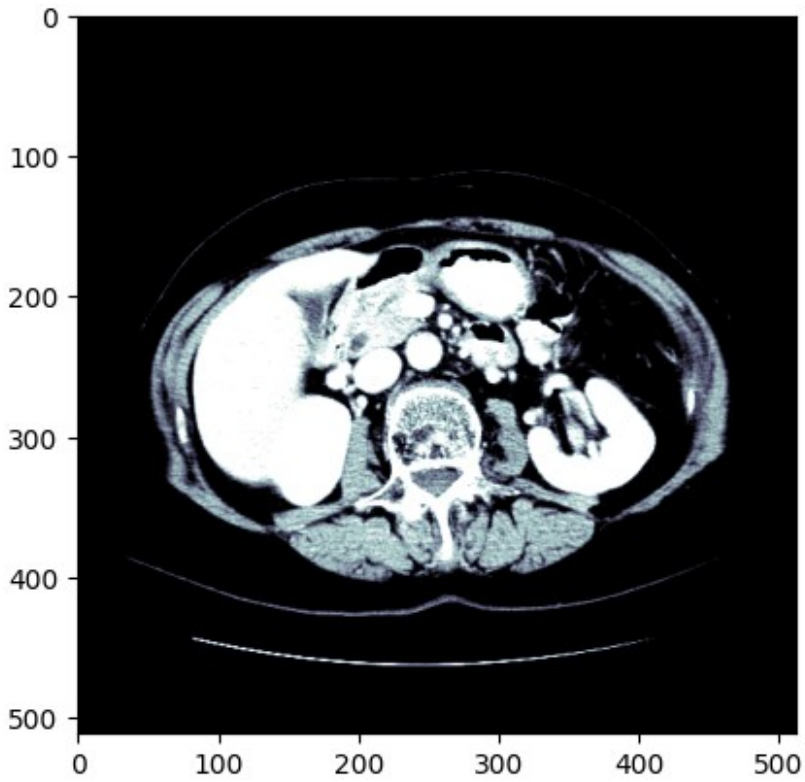
```
# Read sample
sample = 0
sample_ct   = read_nii(df_files.loc[sample,'dirname']
+"/"+df_files.loc[sample,'filename'])
sample_mask  = read_nii(df_files.loc[sample,'mask_dirname']
+"/"+df_files.loc[sample,'mask_filename'])
print(sample_ct.shape)
print(sample_mask.shape)
print(df_files.loc[sample,'dirname']
+"/"+df_files.loc[sample,'filename'])
```

```
(512, 512, 75)
(512, 512, 75)
/kaggle/input/liver-tumor-segmentation/volume_pt1/volume-0.nii
```

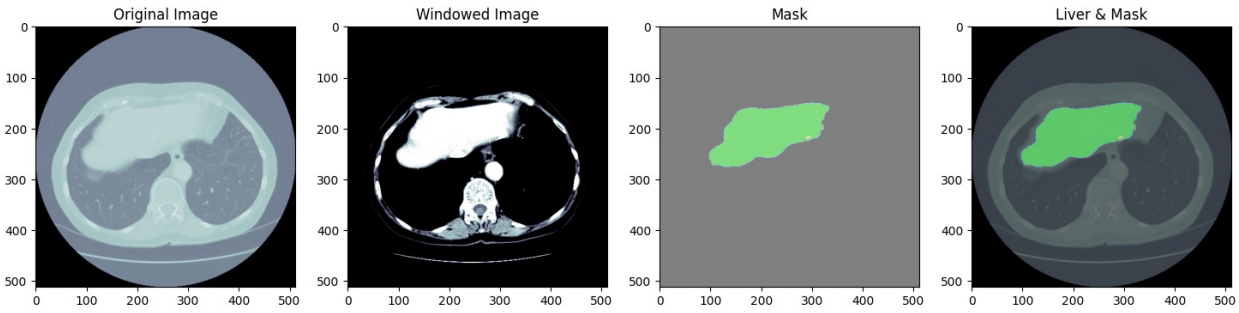
```
print(np.amin(sample_ct), np.amax(sample_ct))
print(np.amin(sample_mask), np.amax(sample_mask))
```

```
-3024.0 1410.0
0.0 2.0
```

```
# Preprocess the nii file
# Source https://docs.fast.ai/medical.imaging

dicom_windows = types.SimpleNamespace(
    brain=(80,40),
    subdural=(254,100),
    stroke=(8,32),
    brain_bone=(2800,600),
    brain_soft=(375,40),
    lungs=(1500,-600),
    mediastinum=(350,50),
    abdomen_soft=(400,50),
    liver=(150,30),
    spine_soft=(250,50),
    spine_bone=(1800,400),
    custom = (200,60)
)

@patch
def windowed(self:Tensor, w, l):
    px = self.clone()
    px_min = l - w//2
    px_max = l + w//2
    px[px<px_min] = px_min
    px[px>px_max] = px_max
    return (px-px_min) / (px_max-px_min)

plt.imshow(tensor(sample_ct[...,50].astype(np.float32)).windowed(*dico
m_windows.liver), cmap=plt.cm.bone);
```




```
def plot_sample(array_list, color_map = 'nipy_spectral'):
    #Plots and a slice with all available annotations
    fig = plt.figure(figsize=(18,15))
    plt.subplot(1,4,1)
    plt.imshow(array_list[0], cmap='bone')
    plt.title('Original Image')
    plt.subplot(1,4,2)

plt.imshow(tensor(array_list[0].astype(np.float32)).windowed(*dicom_wi
ndows.liver), cmap='bone');
    plt.title('Windowed Image')
    plt.subplot(1,4,3)
    plt.imshow(array_list[1], alpha=0.5, cmap=color_map)
    plt.title('Mask')
    plt.subplot(1,4,4)
    plt.imshow(array_list[0], cmap='bone')
    plt.imshow(array_list[1], alpha=0.5, cmap=color_map)
    plt.title('Liver & Mask')
    plt.show()

sample=70
sample_slice = tensor(sample_ct[...,sample].astype(np.float32))

plot_sample([sample_ct[...,sample], sample_mask[...,sample]])
```



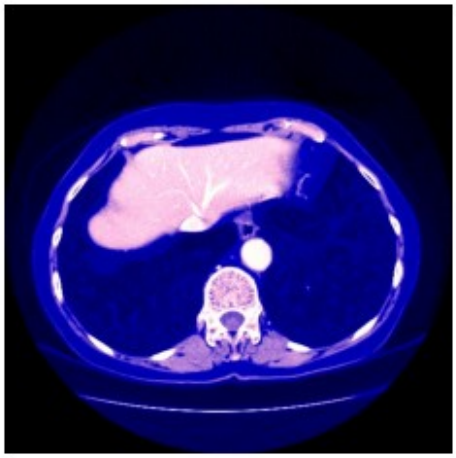

```
# Check the mask values
mask = Image.fromarray(sample_mask[...,sample].astype('uint8'),
mode="L")
unique, counts = np.unique(mask, return_counts=True)
print( np.array((unique, counts)).T)
```

```
[[     0 243918]
 [     1  18193]
 [     2     33]]
```

```
# Preprocessing functions
# Source https://docs.fast.ai/medical.imaging
class TensorCTScan(TensorImageBW): _show_args = {'cmap':'bone'}
@patch
def freqhist_bins(self:Tensor, n_bins=100):
    #A function to split the range of pixel values into groups,such
that each group has around the same number of pixels
    imsd = self.view(-1).sort()[0]
    t = torch.cat([tensor([0.001]),
                   torch.arange(n_bins).float()/n_bins+(1/2/n_bins),
                   tensor([0.999])])
    t = (len(imsd)*t).long()
    return imsd[t].unique()
@patch
def hist_scaled(self:Tensor, brks=None):
    "Scales a tensor using `freqhist_bins` to values between 0 and 1"
    if self.device.type=='cuda': return self.hist_scaled_pt(brks)
    if brks is None: brks = self.freqhist_bins()
    ys = np.linspace(0., 1., len(brks))
    x = self.numpy().flatten()
    x = np.interp(x, brks.numpy(), ys)
    return tensor(x).reshape(self.shape).clamp(0.,1.)
@patch
def to_nchan(x:Tensor, wins, bins=None):
    res = [x.windowed(*win) for win in wins]
    if not isinstance(bins,int) or bins!=0:
res.append(x.hist_scaled(bins).clamp(0,1))
    dim = [0,1][x.dim()==3]
    return TensorCTScan(torch.stack(res, dim=dim))
@patch
```

```
def save_jpg(x:(Tensor), path, wins, bins=None, quality=90):
    fn = Path(path).with_suffix('.jpg')
    x = (x.to_nchan(wins, bins)*255).byte()
    im = Image.fromarray(x.permute(1,2,0).numpy(), mode=['RGB','CMYK']
[x.shape[0]==4])
    im.save(fn, quality=quality)
_,axs=subplots(1,1)
sample_slice.save_jpg('test.jpg',
[dicom_windows.liver,dicom_windows.custom])
show_image(Image.open('test.jpg'), ax=axs[0])
```

```
<Axes: >
```



```
df_files_1=df_files[0:30]
df_files_1
```

```
                                                    dirname
filename  \
0          /kaggle/input/liver-tumor-segmentation/volume_pt1
volume-0.nii
1          /kaggle/input/liver-tumor-segmentation/volume_pt1
volume-1.nii
2          /kaggle/input/liver-tumor-segmentation/volume_pt1   volume-
10.nii
3   /kaggle/input/liver-tumor-segmentation-part-2/volume_pt6  volume-
100.nii
4   /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
101.nii
5   /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
102.nii
6   /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
103.nii
7   /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
104.nii
```

```
8   /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
105.nii
9   /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
106.nii
10  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
107.nii
11  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
108.nii
12  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
109.nii
13         /kaggle/input/liver-tumor-segmentation/volume_pt2   volume-
11.nii
14  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
110.nii
15  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
111.nii
16  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
112.nii
17  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
113.nii
18  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
114.nii
19  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
115.nii
20  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
116.nii
21  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
117.nii
22  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
118.nii
23  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
119.nii
24         /kaggle/input/liver-tumor-segmentation/volume_pt2   volume-
12.nii
25  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
120.nii
26  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
121.nii
27  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
122.nii
28  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
123.nii
29  /kaggle/input/liver-tumor-segmentation-part-2/volume_pt8  volume-
124.nii

                                          mask_dirname
mask_filename
0   ../input/liver-tumor-segmentation/segmentations    segmentation-
0.nii
```

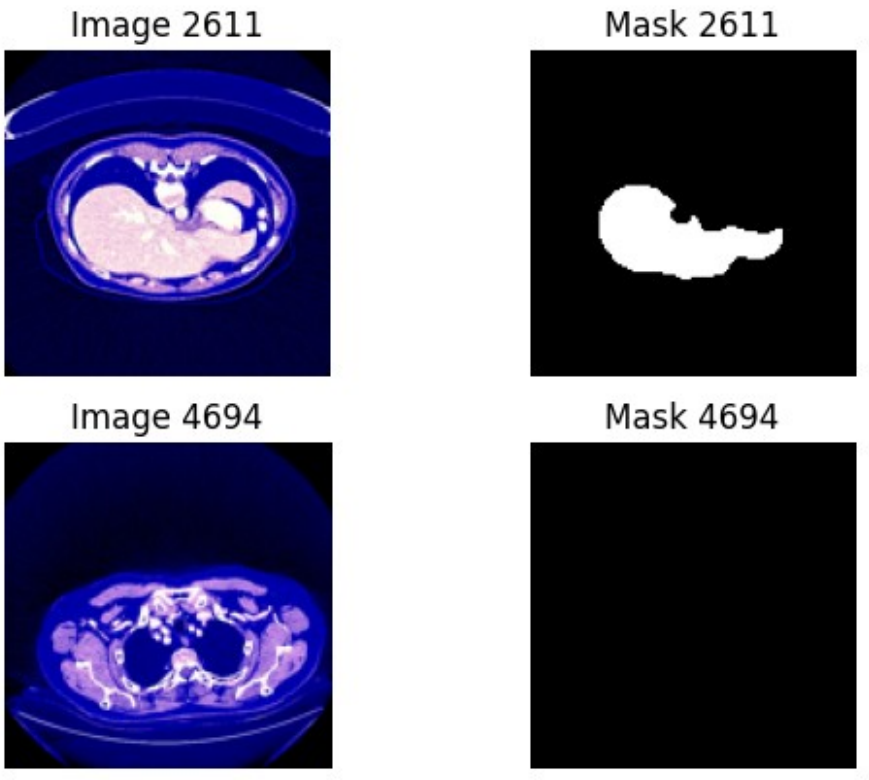
```
1   ../input/liver-tumor-segmentation/segmentations    segmentation-
1.nii
2   ../input/liver-tumor-segmentation/segmentations   segmentation-
10.nii
3   ../input/liver-tumor-segmentation/segmentations  segmentation-
100.nii
4   ../input/liver-tumor-segmentation/segmentations  segmentation-
101.nii
5   ../input/liver-tumor-segmentation/segmentations  segmentation-
102.nii
6   ../input/liver-tumor-segmentation/segmentations  segmentation-
103.nii
7   ../input/liver-tumor-segmentation/segmentations  segmentation-
104.nii
8   ../input/liver-tumor-segmentation/segmentations  segmentation-
105.nii
9   ../input/liver-tumor-segmentation/segmentations  segmentation-
106.nii
10  ../input/liver-tumor-segmentation/segmentations  segmentation-
107.nii
11  ../input/liver-tumor-segmentation/segmentations  segmentation-
108.nii
12  ../input/liver-tumor-segmentation/segmentations  segmentation-
109.nii
13  ../input/liver-tumor-segmentation/segmentations   segmentation-
11.nii
14  ../input/liver-tumor-segmentation/segmentations  segmentation-
110.nii
15  ../input/liver-tumor-segmentation/segmentations  segmentation-
111.nii
16  ../input/liver-tumor-segmentation/segmentations  segmentation-
112.nii
17  ../input/liver-tumor-segmentation/segmentations  segmentation-
113.nii
18  ../input/liver-tumor-segmentation/segmentations  segmentation-
114.nii
19  ../input/liver-tumor-segmentation/segmentations  segmentation-
115.nii
20  ../input/liver-tumor-segmentation/segmentations  segmentation-
116.nii
21  ../input/liver-tumor-segmentation/segmentations  segmentation-
117.nii
22  ../input/liver-tumor-segmentation/segmentations  segmentation-
118.nii
23  ../input/liver-tumor-segmentation/segmentations  segmentation-
119.nii
24  ../input/liver-tumor-segmentation/segmentations   segmentation-
12.nii
25  ../input/liver-tumor-segmentation/segmentations  segmentation-
```

```
120.nii
26  ../input/liver-tumor-segmentation/segmentations  segmentation-
121.nii
27  ../input/liver-tumor-segmentation/segmentations  segmentation-
122.nii
28  ../input/liver-tumor-segmentation/segmentations  segmentation-
123.nii
29  ../input/liver-tumor-segmentation/segmentations  segmentation-
124.nii

#!pip install --upgrade jupyter ipywidgets
#!jupyter nbextension enable --py widgetsnbextension
#!pip install --upgrade pip

import random
GENERATE_JPG_FILES = True
if GENERATE_JPG_FILES:
    path = Path(".")
    os.makedirs('train_images', exist_ok=True)
    os.makedirs('train_masks', exist_ok=True)# Randomly select 1/3 of
the indices
    random_indices = random.sample(range(len(df_files)), len(df_files)
// 2)
    for ii in tqdm(range(len(df_files))):  # take 1/3 nii files for
training
        curr_ct = read_nii(df_files.loc[ii, 'dirname'] + "/" +
df_files.loc[ii, 'filename'])
        curr_mask = read_nii(df_files.loc[ii, 'mask_dirname'] + "/" +
df_files.loc[ii, 'mask_filename'])
        curr_file_name = str(df_files.loc[ii, 'filename']).split('.')
[0]
        curr_dim = curr_ct.shape[2]  # 512, 512, curr_dim
        for curr_slice in range(0, curr_dim, 2):  # export every 2nd
slice for training
            data = tensor(curr_ct[..., curr_slice].astype(np.float32))
            mask = Image.fromarray(curr_mask[...,
curr_slice].astype('uint8'), mode="L")

data.save_jpg(f"train_images/{curr_file_name}_slice_{curr_slice}.jpg",
[dicom_windows.liver, dicom_windows.custom])

mask.save(f"train_masks/{curr_file_name}_slice_{curr_slice}_mask.png")
else:
    path = Path("../input/liver-segmentation-with-fastai-v2")  # read
jpg from saved kernel output

{"model_id":"8bd1fda7e0314b1a9e2fbb0eb82d6542","version_major":2,"vers
ion_minor":0}
```

```
pixdim[0] (qfac) should be 1 (default) or -1; setting qfac to 1
pixdim[0] (qfac) should be 1 (default) or -1; setting qfac to 1
pixdim[0] (qfac) should be 1 (default) or -1; setting qfac to 1
pixdim[0] (qfac) should be 1 (default) or -1; setting qfac to 1
pixdim[0] (qfac) should be 1 (default) or -1; setting qfac to 1
```

```
import os
import random
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from PIL import Image
from sklearn.model_selection import train_test_split
# Set the directories
images_dir = 'train_images'
masks_dir = 'train_masks'
# Get the list of image file names
all_image_file_names = sorted(os.listdir(images_dir))
# Randomly select 9000 images
selected_image_file_names = random.sample(all_image_file_names, 9000)
# Load selected images and masks with resizing
target_size = (128, 128)
images = [np.array(Image.open(os.path.join(images_dir,
fname)).resize(target_size)) for fname in selected_image_file_names]
# Adjust file extension for masks
mask_file_names = [fname.replace('.jpg', '_mask.png') for fname in
selected_image_file_names]
masks = [np.array(Image.open(os.path.join(masks_dir,
fname)).resize(target_size)) for fname in mask_file_names]
# Convert the lists to numpy arrays
images = np.array(images)
masks = np.array(masks)
# Normalize images to [0, 1]
images = images / 255.0
# Reshape masks to include a channel dimension
masks = masks.reshape((masks.shape[0], masks.shape[1], masks.shape[2],
1))

# Split the data into training, validation, and test sets
# 70% training, 20% validation, 10% test
x_train, x_temp, y_train, y_temp = train_test_split(images, masks,
test_size=0.3, random_state=42)
x_valid, x_test, y_valid, y_test = train_test_split(x_temp, y_temp,
test_size=1/3, random_state=42)

import matplotlib.pyplot as plt

# Function to display images and masks
def display_images_and_masks(images, masks, num_samples=2):
```

```
    # Randomly select samples
    sample_indices = random.sample(range(images.shape[0]),
num_samples)

    for i in range(num_samples):
        index = sample_indices[i]

        # Display the image
        plt.subplot(num_samples, 2, 2 * i + 1)
        plt.imshow(images[index])
        plt.title("Image {}".format(index))
        plt.axis("off")

        # Display the mask
        plt.subplot(num_samples, 2, 2 * i + 2)
        plt.imshow(masks[index].squeeze(), cmap='gray')
        plt.title("Mask {}".format(index))
        plt.axis("off")

    plt.show()

# Display two random images and masks
display_images_and_masks(x_train, y_train)
```




```
y_train.shape, x_train.shape
```

```
((6300, 128, 128, 1), (6300, 128, 128, 3))
```

```
from tensorflow import keras
from tensorflow.keras import layers

def create_fcn_model(input_shape=(128, 128, 3), target_shape=(128,
128, 1)):
    model = keras.Sequential()

    # Encoder
    model.add(layers.Conv2D(64, (3, 3), activation='relu',
padding='same', input_shape=input_shape))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(64, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(128, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(128, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D((2, 2)))

    # Decoder with Skip Connections
    model.add(layers.UpSampling2D((2, 2)))
    model.add(layers.Conv2D(128, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())

    model.add(layers.UpSampling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())

    model.add(layers.UpSampling2D((2, 2)))
    model.add(layers.Conv2D(32, (3, 3), activation='relu',
padding='same'))
    model.add(layers.BatchNormalization())
```

```
    # Output layer
    model.add(layers.Conv2D(1, (1, 1), activation='sigmoid'))  #
Assuming binary classification

    # Crop to match the target shape
    crop_height = (model.output_shape[1] - target_shape[0]) // 2
    crop_width = (model.output_shape[2] - target_shape[1]) // 2
    model.add(layers.Cropping2D(cropping=((crop_height, crop_height),
(crop_width, crop_width))))

    return model
# Create the FCN model
model = create_fcn_model()

# Display the model summary
model.summary()
model.save('fcn_model.h5')
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 64)      1792

 batch_normalization (Batch  (None, 128, 128, 64)      256
 Normalization)

 conv2d_1 (Conv2D)           (None, 128, 128, 64)      36928

 batch_normalization_1 (Bat  (None, 128, 128, 64)      256
 chNormalization)

 max_pooling2d (MaxPooling2  (None, 64, 64, 64)        0
 D)

 conv2d_2 (Conv2D)           (None, 64, 64, 128)       73856

 batch_normalization_2 (Bat  (None, 64, 64, 128)       512
 chNormalization)

 conv2d_3 (Conv2D)           (None, 64, 64, 128)       147584

 batch_normalization_3 (Bat  (None, 64, 64, 128)       512
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 32, 32, 128)       0
 g2D)

 conv2d_4 (Conv2D)           (None, 32, 32, 256)       295168
```

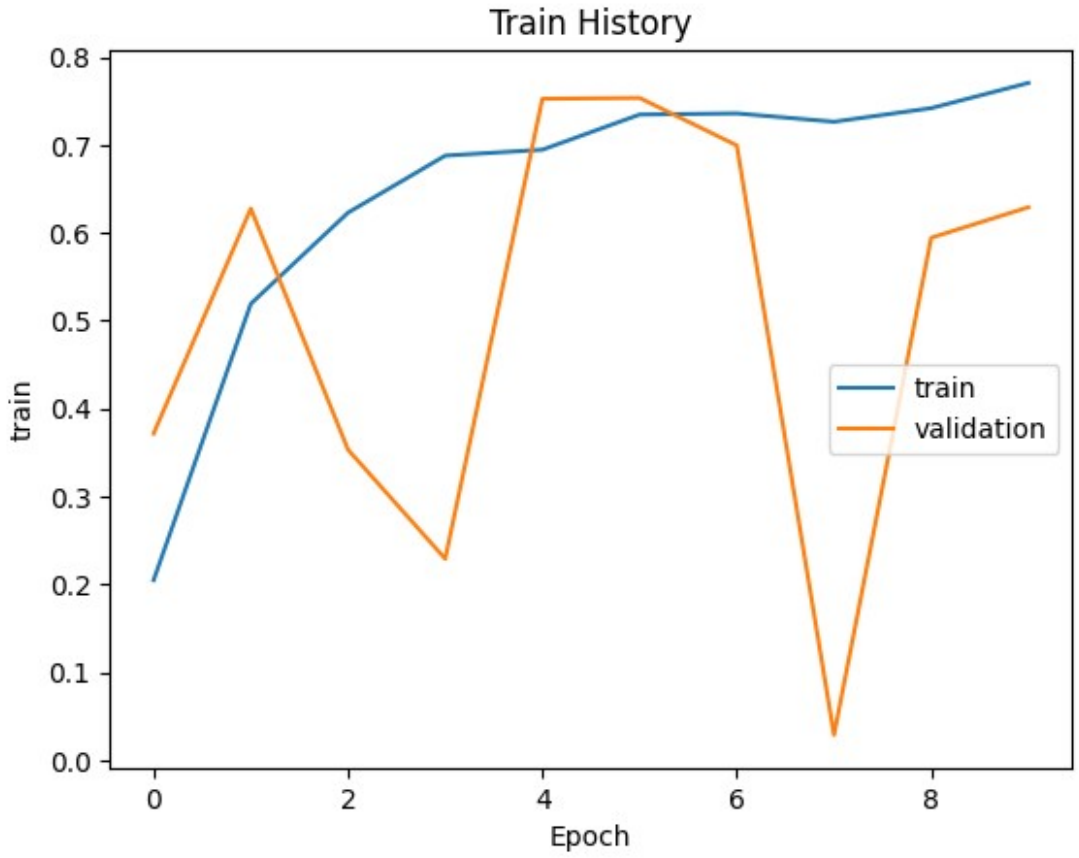
```
 batch_normalization_4 (Bat  (None, 32, 32, 256)       1024
 chNormalization)

 conv2d_5 (Conv2D)           (None, 32, 32, 256)       590080

 batch_normalization_5 (Bat  (None, 32, 32, 256)       1024
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 16, 16, 256)       0
 g2D)

 up_sampling2d (UpSampling2  (None, 32, 32, 256)       0
 D)

 conv2d_6 (Conv2D)           (None, 32, 32, 128)       295040

 batch_normalization_6 (Bat  (None, 32, 32, 128)       512
 chNormalization)

 up_sampling2d_1 (UpSamplin  (None, 64, 64, 128)       0
 g2D)

 conv2d_7 (Conv2D)           (None, 64, 64, 64)        73792

 batch_normalization_7 (Bat  (None, 64, 64, 64)        256
 chNormalization)

 up_sampling2d_2 (UpSamplin  (None, 128, 128, 64)      0
 g2D)

 conv2d_8 (Conv2D)           (None, 128, 128, 32)      18464

 batch_normalization_8 (Bat  (None, 128, 128, 32)      128
 chNormalization)

 conv2d_9 (Conv2D)           (None, 128, 128, 1)       33

 cropping2d (Cropping2D)     (None, 128, 128, 1)       0

=================================================================
Total params: 1537217 (5.86 MB)
Trainable params: 1534977 (5.86 MB)
Non-trainable params: 2240 (8.75 KB)
_________________________________________________________________

/opt/conda/lib/python3.10/site-packages/keras/src/engine/
training.py:3000: UserWarning: You are saving your model as an HDF5
file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
```

```
`model.save('my_model.keras')`.
  saving_api.save_model(

import tensorflow as tf
def dice_coefficient(y_true, y_pred):
    smooth = 1.0  # Smoothing factor to avoid division by zero
    intersection = tf.reduce_sum(y_true * y_pred)
    union = tf.reduce_sum(y_true) + tf.reduce_sum(y_pred)
    dice = (2.0 * intersection + smooth) / (union + smooth)
    return dice

from tensorflow.keras.callbacks import ReduceLROnPlateau

checkpoint = tf.keras.callbacks.ModelCheckpoint("fcn_model.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,

save_weights_only=True, mode='auto')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
patience=5, min_lr=1e-6)
tensorboard_callback =
tf.keras.callbacks.TensorBoard(log_dir='./logs', histogram_freq=1)

from tensorflow.keras.optimizers import Adam
# Compile the model [0.001,0001, 6e-e]
model.compile(optimizer=Adam(lr=.001),loss='binary_crossentropy',
              metrics=[dice_coefficient])

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model.save('fcn_model.h5')

Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.2427 -
dice_coefficient: 0.2048
Epoch 1: val_loss improved from inf to 0.06119, saving model to
fcn_model.h5
394/394 [==============================] - 32s 54ms/step - loss:
0.2427 - dice_coefficient: 0.2048 - val_loss: 0.0612 -
val_dice_coefficient: 0.3716 - lr: 0.0010
Epoch 2/10
393/394 [============================>.] - ETA: 0s - loss: 0.0411 -
dice_coefficient: 0.5201
Epoch 2: val_loss improved from 0.06119 to 0.03679, saving model to
fcn_model.h5
394/394 [==============================] - 18s 47ms/step - loss:
0.0411 - dice_coefficient: 0.5194 - val_loss: 0.0368 -
val_dice_coefficient: 0.6278 - lr: 0.0010
Epoch 3/10
393/394 [============================>.] - ETA: 0s - loss: 0.0314 -
```

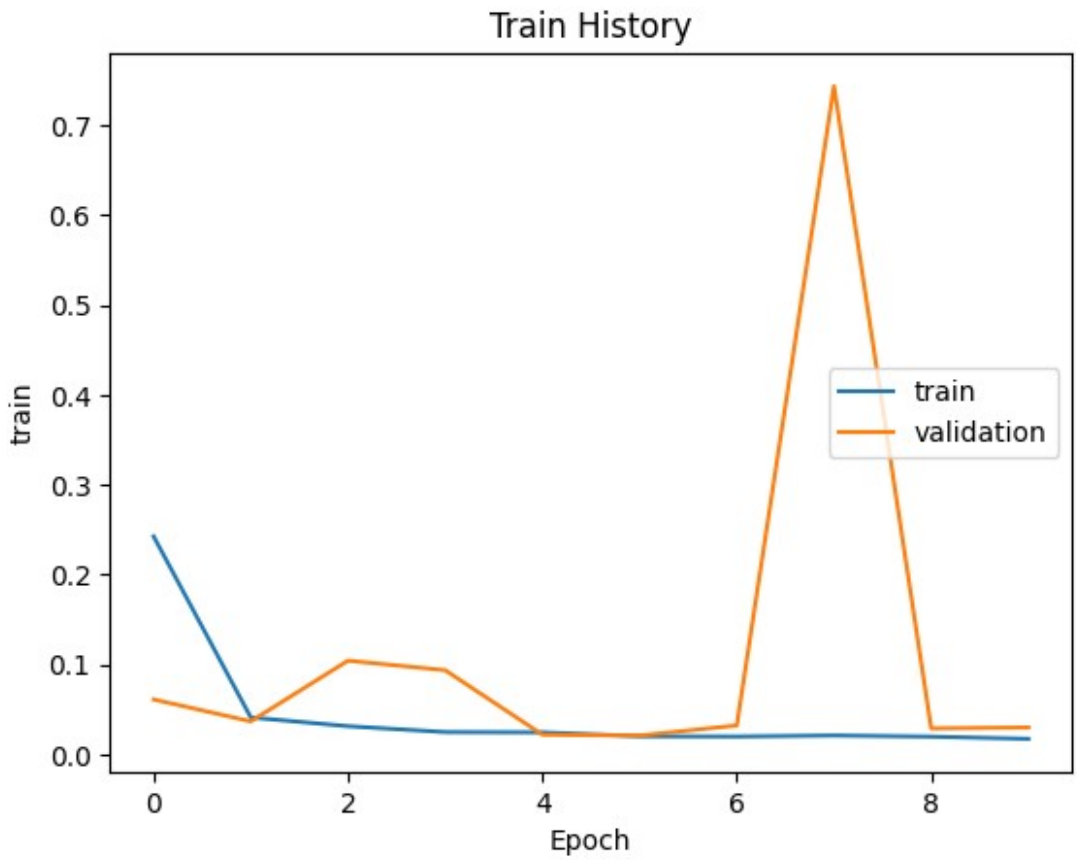
```
dice_coefficient: 0.6231
Epoch 3: val_loss did not improve from 0.03679
394/394 [==============================] - 18s 47ms/step - loss:
0.0314 - dice_coefficient: 0.6231 - val_loss: 0.1044 -
val_dice_coefficient: 0.3533 - lr: 0.0010
Epoch 4/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.6885
Epoch 4: val_loss did not improve from 0.03679
394/394 [==============================] - 18s 46ms/step - loss:
0.0249 - dice_coefficient: 0.6880 - val_loss: 0.0937 -
val_dice_coefficient: 0.2290 - lr: 0.0010
Epoch 5/10
393/394 [============================>.] - ETA: 0s - loss: 0.0245 -
dice_coefficient: 0.6950
Epoch 5: val_loss improved from 0.03679 to 0.02197, saving model to
fcn_model.h5
394/394 [==============================] - 18s 47ms/step - loss:
0.0246 - dice_coefficient: 0.6948 - val_loss: 0.0220 -
val_dice_coefficient: 0.7527 - lr: 0.0010
Epoch 6/10
393/394 [============================>.] - ETA: 0s - loss: 0.0201 -
dice_coefficient: 0.7350
Epoch 6: val_loss improved from 0.02197 to 0.02094, saving model to
fcn_model.h5
394/394 [==============================] - 18s 47ms/step - loss:
0.0202 - dice_coefficient: 0.7348 - val_loss: 0.0209 -
val_dice_coefficient: 0.7537 - lr: 0.0010
Epoch 7/10
393/394 [============================>.] - ETA: 0s - loss: 0.0199 -
dice_coefficient: 0.7361
Epoch 7: val_loss did not improve from 0.02094
394/394 [==============================] - 19s 50ms/step - loss:
0.0199 - dice_coefficient: 0.7361 - val_loss: 0.0323 -
val_dice_coefficient: 0.6997 - lr: 0.0010
Epoch 8/10
393/394 [============================>.] - ETA: 0s - loss: 0.0211 -
dice_coefficient: 0.7265
Epoch 8: val_loss did not improve from 0.02094
394/394 [==============================] - 18s 47ms/step - loss:
0.0211 - dice_coefficient: 0.7266 - val_loss: 0.7439 -
val_dice_coefficient: 0.0285 - lr: 0.0010
Epoch 9/10
393/394 [============================>.] - ETA: 0s - loss: 0.0197 -
dice_coefficient: 0.7419
Epoch 9: val_loss did not improve from 0.02094
394/394 [==============================] - 18s 47ms/step - loss:
0.0198 - dice_coefficient: 0.7421 - val_loss: 0.0290 -
val_dice_coefficient: 0.5945 - lr: 0.0010
```

```
Epoch 10/10
393/394 [============================>.] - ETA: 0s - loss: 0.0173 -
dice_coefficient: 0.7711
Epoch 10: val_loss did not improve from 0.02094
394/394 [==============================] - 18s 47ms/step - loss:
0.0173 - dice_coefficient: 0.7708 - val_loss: 0.0300 -
val_dice_coefficient: 0.6293 - lr: 0.0010
```

```
import matplotlib.pyplot as plt
def show_history(history, train, validation):
    plt.plot(history.history[train])
    plt.plot(history.history[validation])
    plt.title('Train History')
    plt.ylabel('train')
    plt.xlabel('Epoch')
    plt.legend(['train', 'validation'], loc='center right')
    plt.show()
show_history(history, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history, 'loss', 'val_loss')
```




```
scores = model.evaluate(x_valid, y_valid)
scores[1]
```

```
57/57 [==============================] - 2s 26ms/step - loss: 0.0300 -
dice_coefficient: 0.6332
```

```
0.6332106590270996
```

```
prediction = model.predict(x_test)
#print(prediction)
#print(y_test)
```

```
29/29 [==============================] - 1s 34ms/step
```

```
test_scores = model.evaluate(x_test, y_test)
test_scores[1]
```

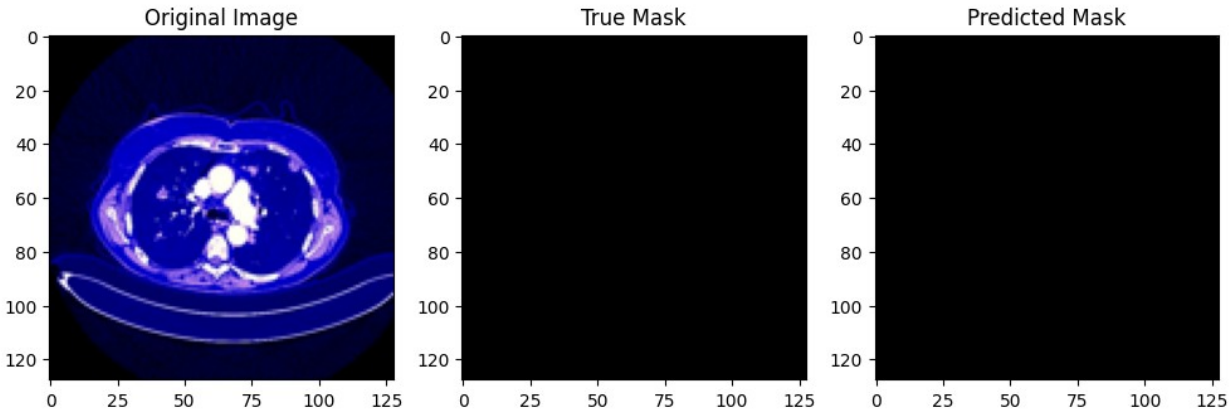
```
29/29 [==============================] - 1s 25ms/step - loss: 0.0355 -
dice_coefficient: 0.6273
```

```
0.6272746920585632
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Assuming you have x_test and y_test
# Replace this with the actual index of the image you want to
visualize
image_index = 44

# Load the image and true mask
input_image = x_test[image_index]
true_mask = y_test[image_index]

# Obtain the predicted mask from model2
predicted_mask = model.predict(np.expand_dims(input_image, axis=0))[0]

# Threshold the predicted mask (assuming it's a binary mask)
threshold = 0.5  # Adjust this threshold based on your model's output
predicted_mask_binary = (predicted_mask > threshold).astype(np.uint8)

# Plotting
plt.figure(figsize=(12, 4))

# Plot original image
plt.subplot(1, 3, 1)
plt.imshow(input_image)
plt.title('Original Image')

# Plot true mask
plt.subplot(1, 3, 2)
plt.imshow(true_mask[:, :, 0], cmap='gray')
plt.title('True Mask')

# Plot predicted mask
plt.subplot(1, 3, 3)
plt.imshow(predicted_mask[:, :, 0], cmap='gray')
plt.title('Predicted Mask')

plt.show()
```

```
1/1 [==============================] - 0s 173ms/step
```



```
import tensorflow.keras.backend as K
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.metrics import MeanIoU
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D, concatenate

# Define Dice coefficient as a metric
def dice_coefficient(y_true, y_pred, smooth=1):
    intersection = K.sum(K.abs(y_true * y_pred), axis=-1)
    union = K.sum(y_true, axis=-1) + K.sum(y_pred, axis=-1)
    return (2. * intersection + smooth) / (union + smooth)

# U-Net model architecture for liver segmentation
def unet_model(input_shape=(128, 128, 3)):
    inputs = Input(input_shape)

    # Encoder
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')
(inputs)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')
(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')
(pool1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')
(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    # Middle
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')
(pool2)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')
(conv3)

    # Decoder
    up4 = UpSampling2D(size=(2, 2))(conv3)
    up4 = Conv2D(64, (3, 3), activation='relu', padding='same')(up4)
    up4 = Conv2D(64, (3, 3), activation='relu', padding='same')(up4)
    merge4 = concatenate([conv2, up4], axis=3)

    up5 = UpSampling2D(size=(2, 2))(merge4)
    up5 = Conv2D(32, (3, 3), activation='relu', padding='same')(up5)
    up5 = Conv2D(32, (3, 3), activation='relu', padding='same')(up5)
    merge5 = concatenate([conv1, up5], axis=3)

    # Output layer
    outputs = Conv2D(1, (1, 1), activation='sigmoid')(merge5)

    model1 = Model(inputs=inputs, outputs=outputs)
```

```
    return model1

# Build the U-Net model
model1 = unet_model()

# Display the model summary
model1.summary()
model1.save('unet_model.h5')
```

```
Model: "model"
__________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==========================================================================================
 input_1 (InputLayer)        [(None, 128, 128, 3)]     0         []


 conv2d_10 (Conv2D)          (None, 128, 128, 32)      896       ['input_1[0][0]']


 conv2d_11 (Conv2D)          (None, 128, 128, 32)      9248      ['conv2d_10[0][0]']


 max_pooling2d_3 (MaxPoolin  (None, 64, 64, 32)        0         ['conv2d_11[0][0]']
 g2D)


 conv2d_12 (Conv2D)          (None, 64, 64, 64)        18496     ['max_pooling2d_3[0][0]']


 conv2d_13 (Conv2D)          (None, 64, 64, 64)        36928     ['conv2d_12[0][0]']


 max_pooling2d_4 (MaxPoolin  (None, 32, 32, 64)        0         ['conv2d_13[0][0]']
 g2D)


 conv2d_14 (Conv2D)          (None, 32, 32, 128)       73856     ['max_pooling2d_4[0][0]']
```

```
 conv2d_15 (Conv2D)          (None, 32, 32, 128)          147584
['conv2d_14[0][0]']


 up_sampling2d_3 (UpSamplin  (None, 64, 64, 128)          0
['conv2d_15[0][0]']
 g2D)



 conv2d_16 (Conv2D)          (None, 64, 64, 64)           73792
['up_sampling2d_3[0][0]']


 conv2d_17 (Conv2D)          (None, 64, 64, 64)           36928
['conv2d_16[0][0]']


 concatenate (Concatenate)   (None, 64, 64, 128)          0
['conv2d_13[0][0]',

'conv2d_17[0][0]']


 up_sampling2d_4 (UpSamplin  (None, 128, 128, 128)        0
['concatenate[0][0]']
 g2D)



 conv2d_18 (Conv2D)          (None, 128, 128, 32)         36896
['up_sampling2d_4[0][0]']


 conv2d_19 (Conv2D)          (None, 128, 128, 32)         9248
['conv2d_18[0][0]']


 concatenate_1 (Concatenate  (None, 128, 128, 64)         0
['conv2d_11[0][0]',
 )
'conv2d_19[0][0]']


 conv2d_20 (Conv2D)          (None, 128, 128, 1)          65
['concatenate_1[0][0]']


==================================================================
```

```
============================
Total params: 443937 (1.69 MB)
Trainable params: 443937 (1.69 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
_____________________________
```

```
from tensorflow.keras.utils import plot_model
#tf.keras.utils.plot_model(model1, show_shapes=True)
# Assuming you have created the model using the create_fcn_model
function
#model1 = create_unet_model()

# Save the model architecture plot to a file
#plot_model(model1, to_file='unet_model.png', show_shapes=False)

checkpoint = tf.keras.callbacks.ModelCheckpoint("unet_model.h5",
monitor='val_loss', verbose=1, patience =
3,save_best_only=True,mode='auto')
import tensorflow as tf
# Define callbacks
checkpoint = tf.keras.callbacks.ModelCheckpoint("unet_model.h5",
monitor='val_loss',verbose=1,patience=3,save_best_only=True,
mode='auto')
# Define other similar callbacks
early_stopping =
tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5,mode='a
uto',restore_best_weights=True)

from tensorflow.keras.optimizers import Adam
# Compile the model [0.001,0001, 6e-e]
model1.compile(optimizer='rmsprop',loss= 'mean_absolute_error',
metrics=[dice_coefficient])

# Train the model
history1 = model1.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid), callbacks=[checkpoint])
model1.save('unet_model.h5')
```

```
Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.0277 -
dice_coefficient: 0.9859
Epoch 1: val_loss improved from inf to 0.02350, saving model to
unet_model.h5
394/394 [==============================] - 20s 40ms/step - loss:
0.0277 - dice_coefficient: 0.9859 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 2/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
```

```
Epoch 2: val_loss improved from 0.02350 to 0.02350, saving model to
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 3/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 3: val_loss improved from 0.02350 to 0.02350, saving model to
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 4/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 4: val_loss improved from 0.02350 to 0.02349, saving model to
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 5/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 5: val_loss improved from 0.02349 to 0.02349, saving model to
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 6/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 6: val_loss improved from 0.02349 to 0.02349, saving model to
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 7/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 7: val_loss improved from 0.02349 to 0.02349, saving model to
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 8/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 8: val_loss improved from 0.02349 to 0.02349, saving model to
```

```
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 9/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 9: val_loss improved from 0.02349 to 0.02349, saving model to
unet_model.h5
394/394 [==============================] - 14s 35ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
Epoch 10/10
393/394 [============================>.] - ETA: 0s - loss: 0.0248 -
dice_coefficient: 0.9880
Epoch 10: val_loss improved from 0.02349 to 0.02349, saving model to
unet_model.h5
394/394 [==============================] - 13s 34ms/step - loss:
0.0248 - dice_coefficient: 0.9880 - val_loss: 0.0235 -
val_dice_coefficient: 0.9887
```

```
scores1 = model1.evaluate(x_valid, y_valid)
scores1[1]
```

```
57/57 [==============================] - 2s 19ms/step - loss: 0.0235 -
dice_coefficient: 0.9887
```

```
0.988682746887207
```

```
prediction1 = model1.predict(x_test)
#print(prediction)
#print(y_test)
```

```
29/29 [==============================] - 1s 24ms/step
```

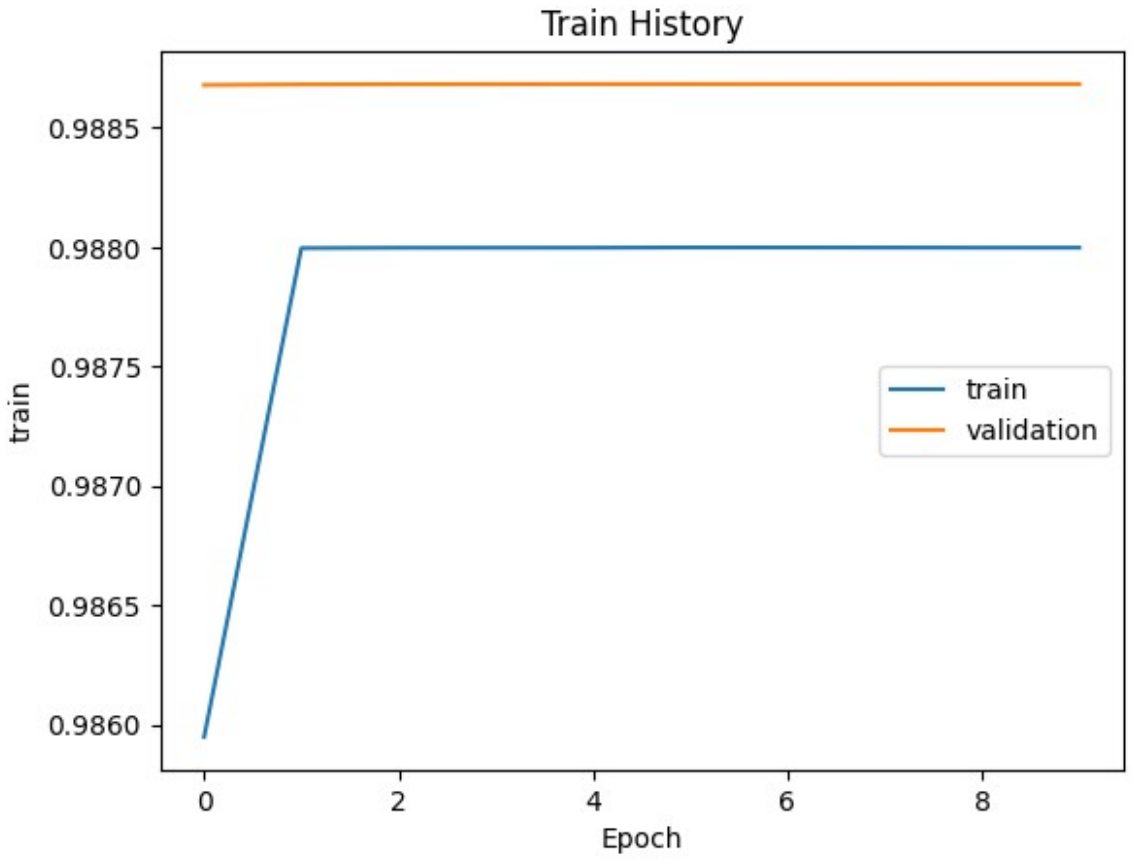
```
test_scores1 = model1.evaluate(x_test, y_test)
test_scores1[1]
```

```
29/29 [==============================] - 1s 19ms/step - loss: 0.0253 -
dice_coefficient: 0.9878
```

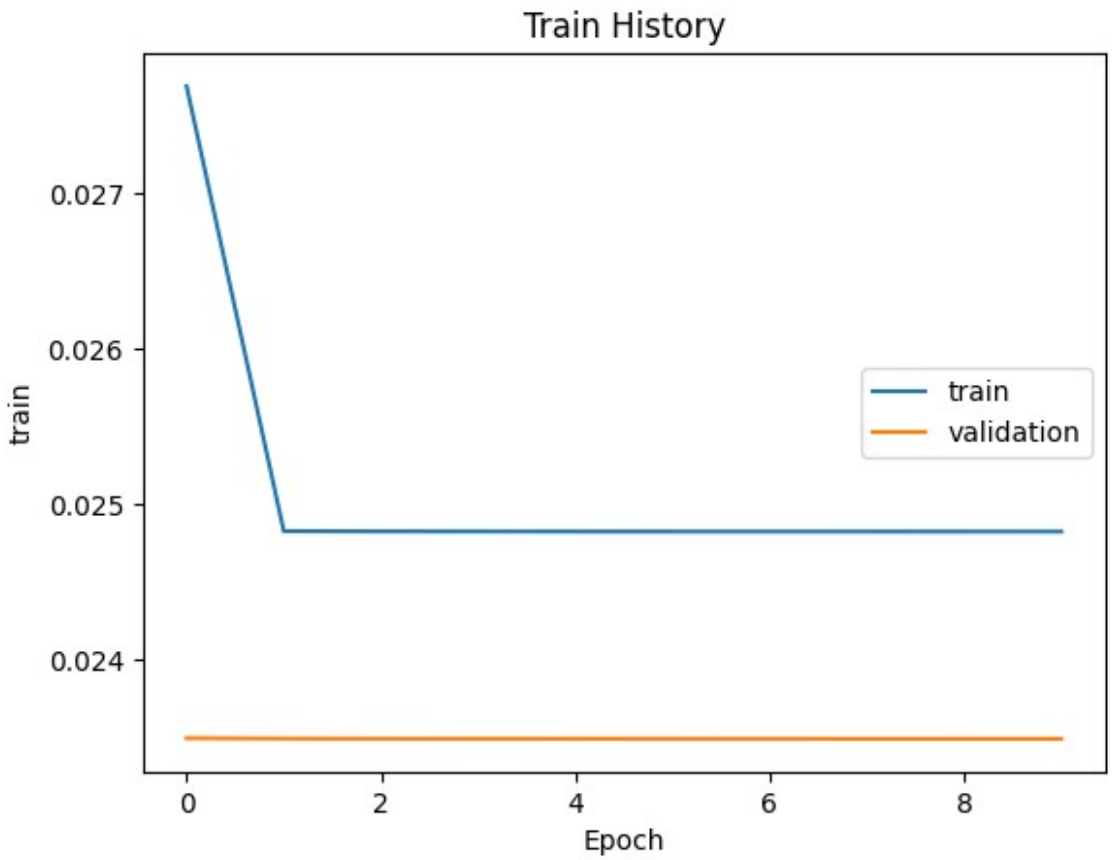
```
0.9878345131874084
```

```
import matplotlib.pyplot as plt
def show_history(history, train, validation,x=1):
    plt.plot(history.history[train])
    plt.plot(history.history[validation])
    plt.title('Train History')
    plt.ylabel('train')
    plt.xlabel('Epoch')
    plt.legend(['train', 'validation'], loc='center right')
```

```
    plt.savefig(f'{train}dice_coefficient_{str(x)}.png')
    plt.show()
show_history(history1, 'dice_coefficient', 'val_dice_coefficient')
```



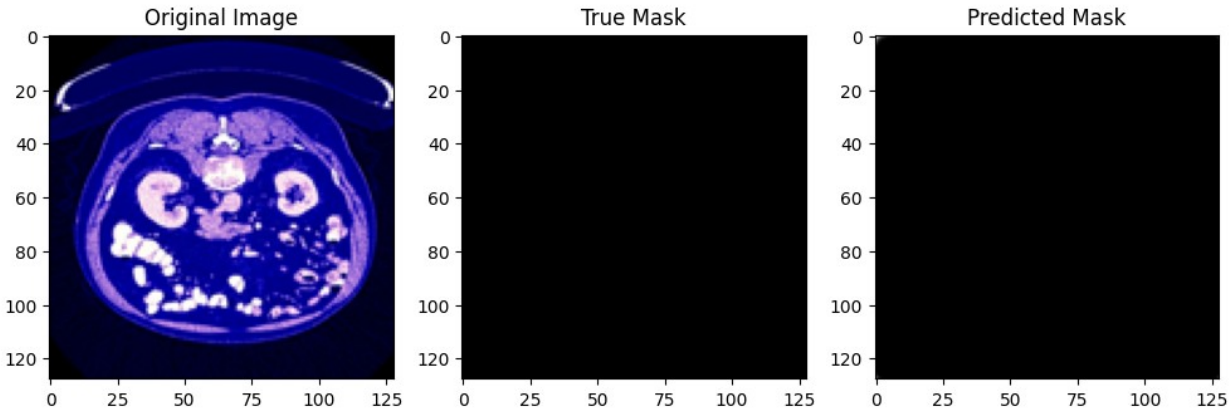

```
show_history(history1, 'loss', 'val_loss')
```



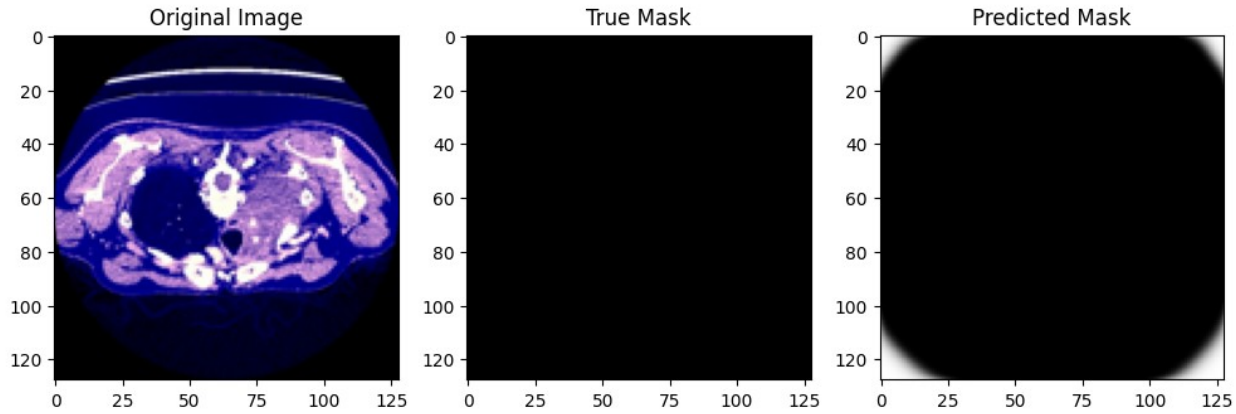

```
import numpy as np
import matplotlib.pyplot as plt
import random

# Generate 10 random values between 1 and 100
random_values = [random.randint(1, 900) for _ in range(10)]

# Assuming you have x_test and y_test
# Replace this with the actual index of the image you want to
visualize
for i in random_values:
    image_index = i

    # Load the image and true mask
    input_image = x_test[image_index]
    true_mask = y_test[image_index]

    # Obtain the predicted mask from model2
    predicted_mask = model1.predict(np.expand_dims(input_image,
axis=0))[0]

    # Threshold the predicted mask (assuming it's a binary mask)
    threshold = 0.5  # Adjust this threshold based on your model's
```

```
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)

    # Plotting
    plt.figure(figsize=(12, 4))

    # Plot original image
    plt.subplot(1, 3, 1)
    plt.imshow(input_image)
    plt.title('Original Image')

    # Plot true mask
    plt.subplot(1, 3, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')

    # Plot predicted mask
    plt.subplot(1, 3, 3)
    plt.imshow(predicted_mask[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')
    plt.show()
```

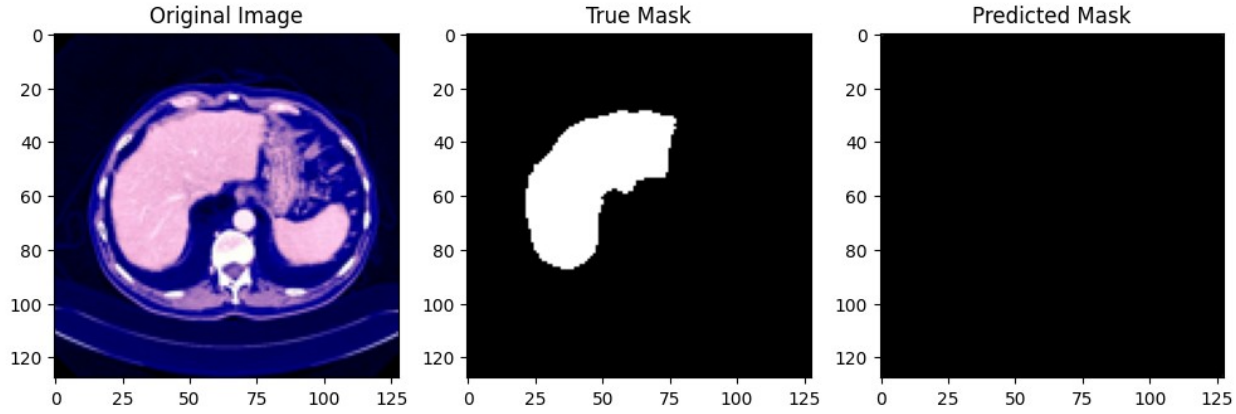
```
1/1 [==============================] - 0s 160ms/step
```



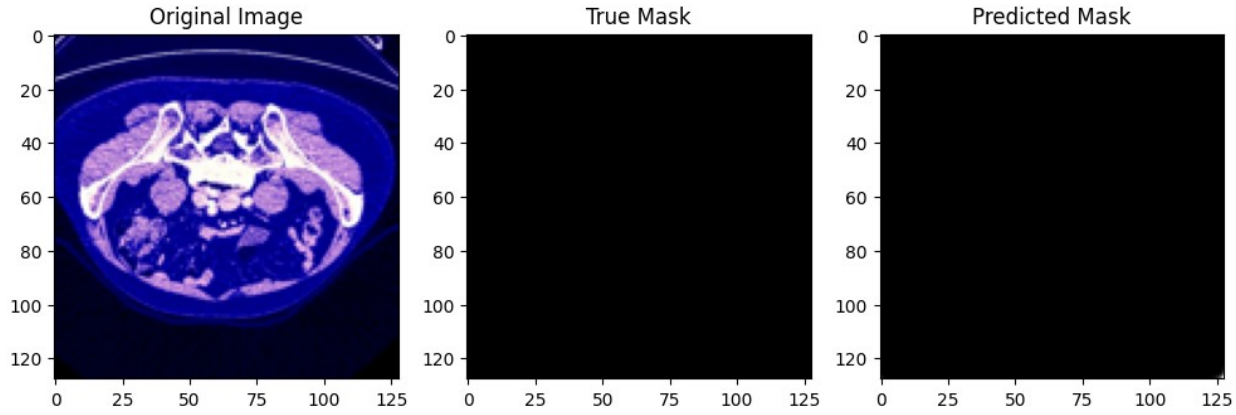

```
1/1 [==============================] - 0s 24ms/step
```



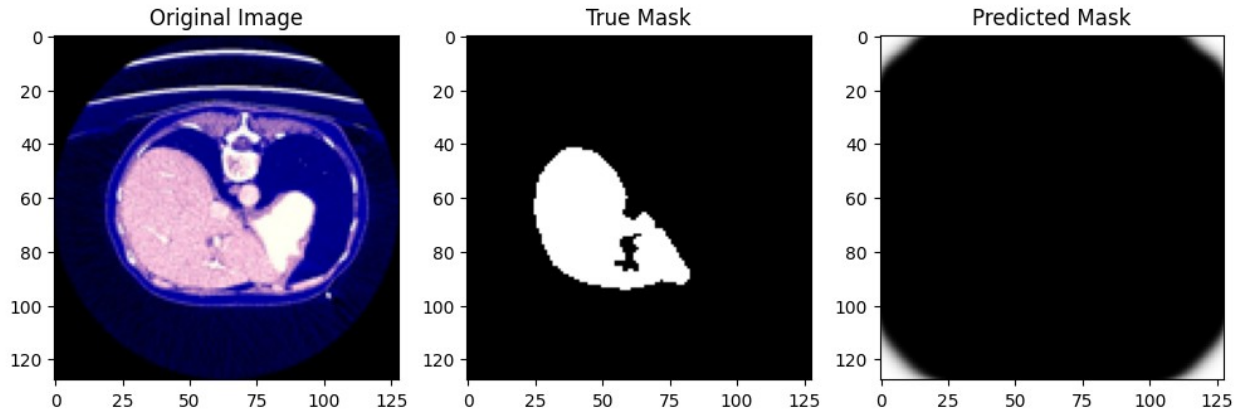

```
1/1 [==============================] - 0s 23ms/step
```



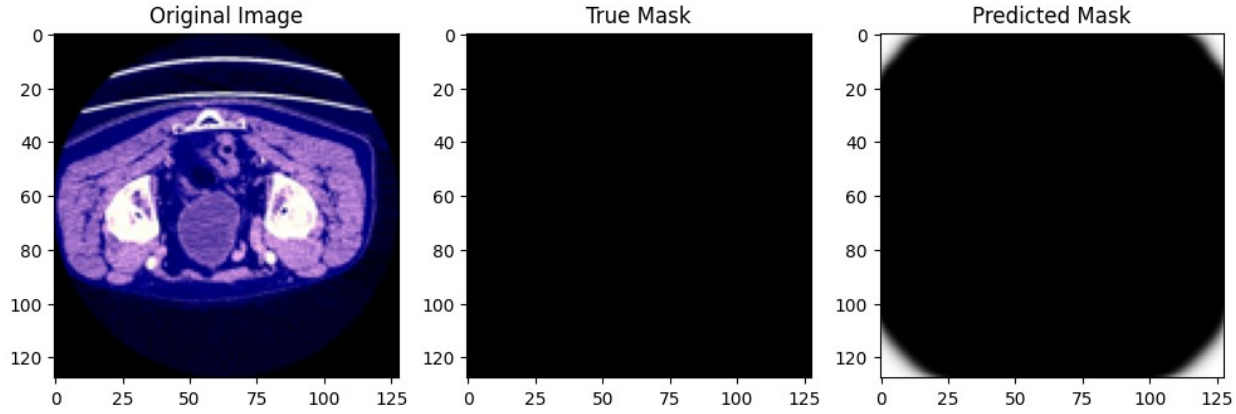

```
1/1 [==============================] - 0s 22ms/step
```



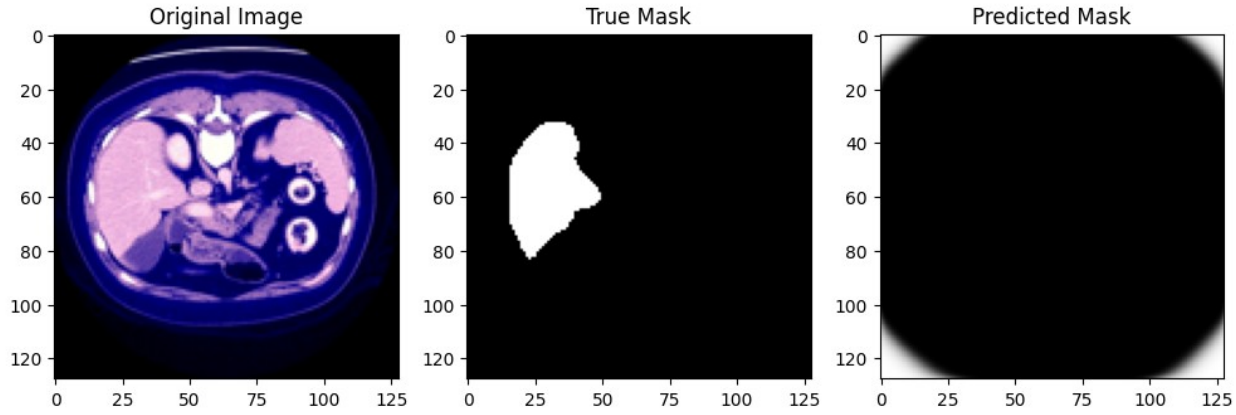

```
1/1 [==============================] - 0s 23ms/step
```



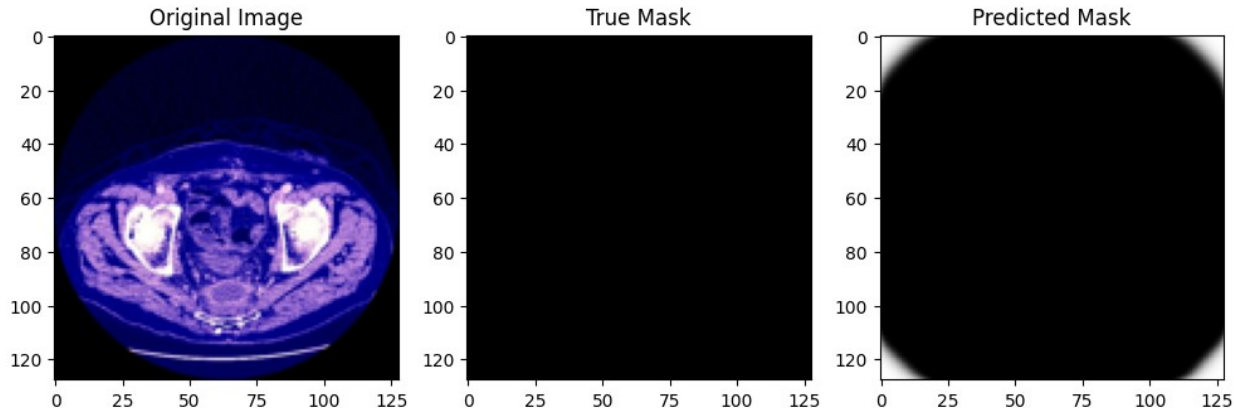

```
1/1 [==============================] - 0s 22ms/step
```



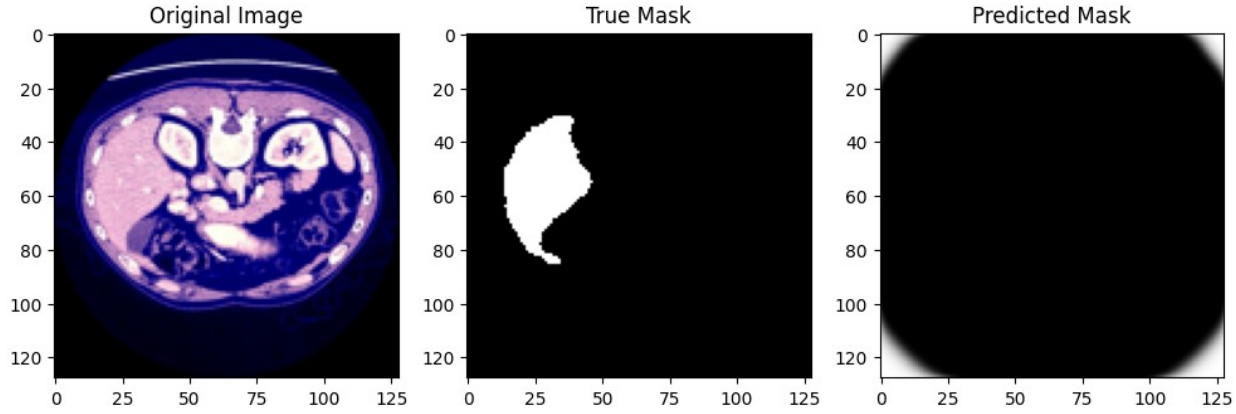

```
1/1 [==============================] - 0s 24ms/step
```



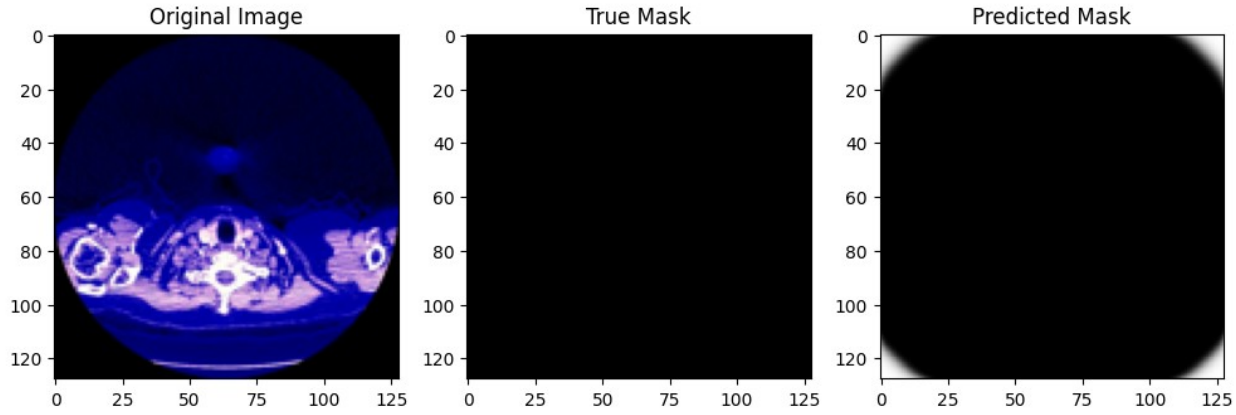

```
1/1 [==============================] - 0s 24ms/step
```




```
1/1 [==============================] - 0s 21ms/step
```




```
1/1 [==============================] - 0s 23ms/step
```




```
def segnet_model(input_shape=(128, 128, 3)):
    model2 = tf.keras.Sequential()
    # Encoder
    model2.add(layers.Conv2D(64, (3, 3), padding='same',
input_shape=input_shape))
```

```
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.Conv2D(64, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model2.add(layers.Conv2D(128, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.Conv2D(128, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model2.add(layers.Conv2D(256, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.Conv2D(256, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.Conv2D(256, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    # Decoder
    model2.add(layers.UpSampling2D(size=(2, 2)))
    model2.add(layers.Conv2D(256, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.Conv2D(256, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.Conv2D(256, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))

    model2.add(layers.UpSampling2D(size=(2, 2)))
    model2.add(layers.Conv2D(128, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
    model2.add(layers.Conv2D(128, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))

    model2.add(layers.UpSampling2D(size=(2, 2)))
    model2.add(layers.Conv2D(64, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))
```

```
    model2.add(layers.Conv2D(64, (3, 3), padding='same'))
    model2.add(layers.BatchNormalization())
    model2.add(layers.Activation('relu'))

    # Output layer
    model2.add(layers.Conv2D(1, (1, 1), activation='sigmoid',
padding='same'))

    return model2

# Create SegNet model
model2 = segnet_model(input_shape=(128, 128, 3))


# Display the model summary
model2.summary()
model2.save('segnet_model2.h5')
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_21 (Conv2D)          (None, 128, 128, 64)      1792

 batch_normalization_9 (Bat  (None, 128, 128, 64)      256
 chNormalization)

 activation (Activation)     (None, 128, 128, 64)      0

 conv2d_22 (Conv2D)          (None, 128, 128, 64)      36928

 batch_normalization_10 (Ba  (None, 128, 128, 64)      256
 tchNormalization)

 activation_1 (Activation)   (None, 128, 128, 64)      0

 max_pooling2d_5 (MaxPoolin  (None, 64, 64, 64)        0
 g2D)

 conv2d_23 (Conv2D)          (None, 64, 64, 128)       73856

 batch_normalization_11 (Ba  (None, 64, 64, 128)       512
 tchNormalization)

 activation_2 (Activation)   (None, 64, 64, 128)       0

 conv2d_24 (Conv2D)          (None, 64, 64, 128)       147584

 batch_normalization_12 (Ba  (None, 64, 64, 128)       512
 tchNormalization)
```

```
activation_3 (Activation)   (None, 64, 64, 128)       0

max_pooling2d_6 (MaxPoolin  (None, 32, 32, 128)       0
g2D)

conv2d_25 (Conv2D)          (None, 32, 32, 256)       295168

batch_normalization_13 (Ba  (None, 32, 32, 256)       1024
tchNormalization)

activation_4 (Activation)   (None, 32, 32, 256)       0

conv2d_26 (Conv2D)          (None, 32, 32, 256)       590080

batch_normalization_14 (Ba  (None, 32, 32, 256)       1024
tchNormalization)

activation_5 (Activation)   (None, 32, 32, 256)       0

conv2d_27 (Conv2D)          (None, 32, 32, 256)       590080

batch_normalization_15 (Ba  (None, 32, 32, 256)       1024
tchNormalization)

activation_6 (Activation)   (None, 32, 32, 256)       0

max_pooling2d_7 (MaxPoolin  (None, 16, 16, 256)       0
g2D)

up_sampling2d_5 (UpSamplin  (None, 32, 32, 256)       0
g2D)

conv2d_28 (Conv2D)          (None, 32, 32, 256)       590080

batch_normalization_16 (Ba  (None, 32, 32, 256)       1024
tchNormalization)

activation_7 (Activation)   (None, 32, 32, 256)       0

conv2d_29 (Conv2D)          (None, 32, 32, 256)       590080

batch_normalization_17 (Ba  (None, 32, 32, 256)       1024
tchNormalization)

activation_8 (Activation)   (None, 32, 32, 256)       0

conv2d_30 (Conv2D)          (None, 32, 32, 256)       590080
```

```
 batch_normalization_18 (Ba  (None, 32, 32, 256)       1024
 tchNormalization)

 activation_9 (Activation)   (None, 32, 32, 256)       0

 up_sampling2d_6 (UpSamplin  (None, 64, 64, 256)       0
 g2D)

 conv2d_31 (Conv2D)          (None, 64, 64, 128)       295040

 batch_normalization_19 (Ba  (None, 64, 64, 128)       512
 tchNormalization)

 activation_10 (Activation)  (None, 64, 64, 128)       0

 conv2d_32 (Conv2D)          (None, 64, 64, 128)       147584

 batch_normalization_20 (Ba  (None, 64, 64, 128)       512
 tchNormalization)

 activation_11 (Activation)  (None, 64, 64, 128)       0

 up_sampling2d_7 (UpSamplin  (None, 128, 128, 128)     0
 g2D)

 conv2d_33 (Conv2D)          (None, 128, 128, 64)      73792

 batch_normalization_21 (Ba  (None, 128, 128, 64)      256
 tchNormalization)

 activation_12 (Activation)  (None, 128, 128, 64)      0

 conv2d_34 (Conv2D)          (None, 128, 128, 64)      36928

 batch_normalization_22 (Ba  (None, 128, 128, 64)      256
 tchNormalization)

 activation_13 (Activation)  (None, 128, 128, 64)      0

 conv2d_35 (Conv2D)          (None, 128, 128, 1)       65

=================================================================
Total params: 4068353 (15.52 MB)
Trainable params: 4063745 (15.50 MB)
Non-trainable params: 4608 (18.00 KB)
_________________________________________________________________
```

```
from tensorflow.keras.utils import plot_model
#tf.keras.utils.plot_model(model1, show_shapes=True)
#model1 = create_unet_model()
```

```
# Save the model architecture plot to a file
#plot_model(model2, to_file='segnet_model.png', show_shapes=False)

def precision_score(y_true, y_pred, smooth=1):
    intersection = K.sum(K.abs(y_true * y_pred), axis=-1)
    union = K.sum(y_true, axis=-1) + K.sum(y_pred, axis=-1)
    return (2. * intersection + smooth) / (union + smooth)

def precision_score_(groundtruth_mask, pred_mask):
    true_positive = np.sum(pred_mask * groundtruth_mask)
    false_positive = np.sum(pred_mask * (1 - groundtruth_mask))
    precision = true_positive / (true_positive + false_positive + 1e-
9)  # Adding epsilon to avoid division by zero
    return round(precision, 3)

def recall_score_(groundtruth_mask, pred_mask):
    true_positive = np.sum(pred_mask * groundtruth_mask)
    false_negative = np.sum((1 - pred_mask) * groundtruth_mask)
    recall = true_positive / (true_positive + false_negative + 1e-9)
# Adding epsilon to avoid division by zero
    return round(recall, 3)
def accuracy(groundtruth_mask, pred_mask):
    intersect = np.sum(pred_mask*groundtruth_mask)
    union = np.sum(pred_mask) + np.sum(groundtruth_mask) - intersect
    xor = np.sum(groundtruth_mask==pred_mask)
    acc = np.mean(xor/(union + xor - intersect))
    return round(acc, 3)
def iou(groundtruth_mask, pred_mask):
    intersect = np.sum(pred_mask*groundtruth_mask)
    union = np.sum(pred_mask) + np.sum(groundtruth_mask) - intersect
    iou = np.mean(intersect/union)
    return round(iou, 3)

checkpoint = tf.keras.callbacks.ModelCheckpoint("segnet_model2.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')
checkpoint = tf.keras.callbacks.ModelCheckpoint("segnet_model2.h5",
                                                monitor='val_loss',
                                                verbose=1,
                                                patience=3,
                                                save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                  patience=5,
                                                  mode='auto',

restore_best_weights=True)
```

```
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                 factor=0.2,
                                                 patience=2,
                                                 min_lr=1e-6,
                                                 mode='auto',
                                                 verbose=1)

# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]

from tensorflow.keras.optimizers import Adam
# Compile the model [0.001,0001, 6e-e]
model2.compile(optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999),
               loss= 'binary_crossentropy', metrics=[dice_coefficient])

model2.compile(optimizer='SGD',loss= 'binary_crossentropy',
metrics=[dice_coefficient])

# Train the model
history2 = model2.fit(x_train, y_train, epochs=10,
batch_size=16,validation_data=(x_valid, y_valid),
callbacks=[checkpoint])
model2.save('segnet_model2.h5')
```

```
Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.1346 -
dice_coefficient: 0.9220
Epoch 1: val_loss improved from inf to 0.06186, saving model to
segnet_model2.h5
394/394 [==============================] - 45s 99ms/step - loss:
0.1346 - dice_coefficient: 0.9220 - val_loss: 0.0619 -
val_dice_coefficient: 0.9691
Epoch 2/10
394/394 [==============================] - ETA: 0s - loss: 0.0305 -
dice_coefficient: 0.9774
Epoch 2: val_loss improved from 0.06186 to 0.01806, saving model to
segnet_model2.h5
394/394 [==============================] - 36s 91ms/step - loss:
0.0305 - dice_coefficient: 0.9774 - val_loss: 0.0181 -
val_dice_coefficient: 0.9844
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: 0.0128 -
dice_coefficient: 0.9843
Epoch 3: val_loss improved from 0.01806 to -0.00023, saving model to
segnet_model2.h5
394/394 [==============================] - 36s 92ms/step - loss:
0.0128 - dice_coefficient: 0.9843 - val_loss: -2.3377e-04 -
val_dice_coefficient: 0.9881
Epoch 4/10
394/394 [==============================] - ETA: 0s - loss: 0.0014 -
```

```
dice_coefficient: 0.9866
Epoch 4: val_loss improved from -0.00023 to -0.02369, saving model to
segnet_model2.h5
394/394 [==============================] - 36s 91ms/step - loss:
0.0014 - dice_coefficient: 0.9866 - val_loss: -0.0237 -
val_dice_coefficient: 0.9907
Epoch 5/10
394/394 [==============================] - ETA: 0s - loss: -0.0148 -
dice_coefficient: 0.9888
Epoch 5: val_loss improved from -0.02369 to -0.03082, saving model to
segnet_model2.h5
394/394 [==============================] - 36s 91ms/step - loss: -
0.0148 - dice_coefficient: 0.9888 - val_loss: -0.0308 -
val_dice_coefficient: 0.9903
Epoch 6/10
394/394 [==============================] - ETA: 0s - loss: -0.0304 -
dice_coefficient: 0.9891
Epoch 6: val_loss did not improve from -0.03082
394/394 [==============================] - 36s 90ms/step - loss: -
0.0304 - dice_coefficient: 0.9891 - val_loss: 0.0515 -
val_dice_coefficient: 0.9891
Epoch 7/10
394/394 [==============================] - ETA: 0s - loss: -0.0523 -
dice_coefficient: 0.9907
Epoch 7: val_loss did not improve from -0.03082
394/394 [==============================] - 36s 90ms/step - loss: -
0.0523 - dice_coefficient: 0.9907 - val_loss: 0.0642 -
val_dice_coefficient: 0.9877
Epoch 8/10
394/394 [==============================] - ETA: 0s - loss: -0.0704 -
dice_coefficient: 0.9911
Epoch 8: val_loss improved from -0.03082 to -0.08338, saving model to
segnet_model2.h5
394/394 [==============================] - 36s 91ms/step - loss: -
0.0704 - dice_coefficient: 0.9911 - val_loss: -0.0834 -
val_dice_coefficient: 0.9920
Epoch 9/10
394/394 [==============================] - ETA: 0s - loss: -0.0960 -
dice_coefficient: 0.9920
Epoch 9: val_loss improved from -0.08338 to -0.20497, saving model to
segnet_model2.h5
394/394 [==============================] - 36s 91ms/step - loss: -
0.0960 - dice_coefficient: 0.9920 - val_loss: -0.2050 -
val_dice_coefficient: 0.9911
Epoch 10/10
394/394 [==============================] - ETA: 0s - loss: -0.1299 -
dice_coefficient: 0.9919
Epoch 10: val_loss improved from -0.20497 to -0.28397, saving model to
segnet_model2.h5
```

```
394/394 [==============================] - 36s 91ms/step - loss: -
0.1299 - dice_coefficient: 0.9919 - val_loss: -0.2840 -
val_dice_coefficient: 0.9953
```

```
scores2 = model2.evaluate(x_valid, y_valid)
scores2[1]
```

```
57/57 [==============================] - 5s 55ms/step - loss: -0.2840
- dice_coefficient: 0.9953
```

```
0.9952918887138367
```

```
prediction2 = model2.predict(x_test)
#print(prediction)
#print(y_test)
```

```
29/29 [==============================] - 2s 68ms/step
```

```
test_scores2 = model2.evaluate(x_test, y_test)
test_scores2[1]
```

```
29/29 [==============================] - 2s 53ms/step - loss: -0.3152
- dice_coefficient: 0.9953
```

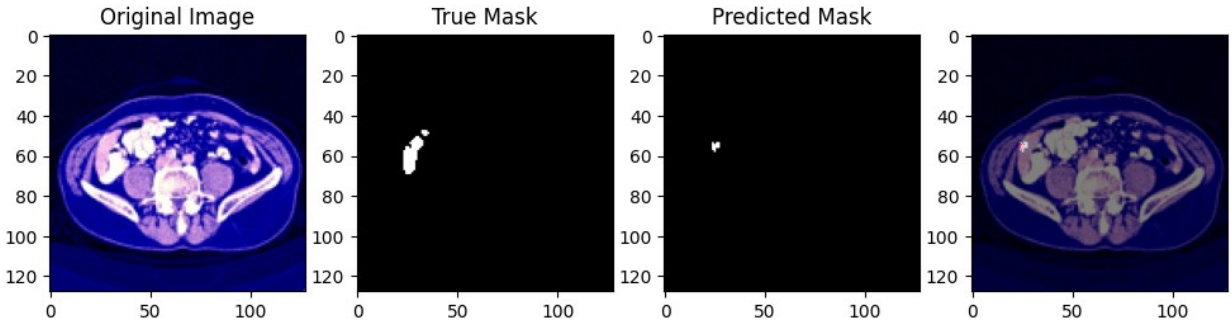
```
0.9952852129936218
```

```
len(x_test)
```

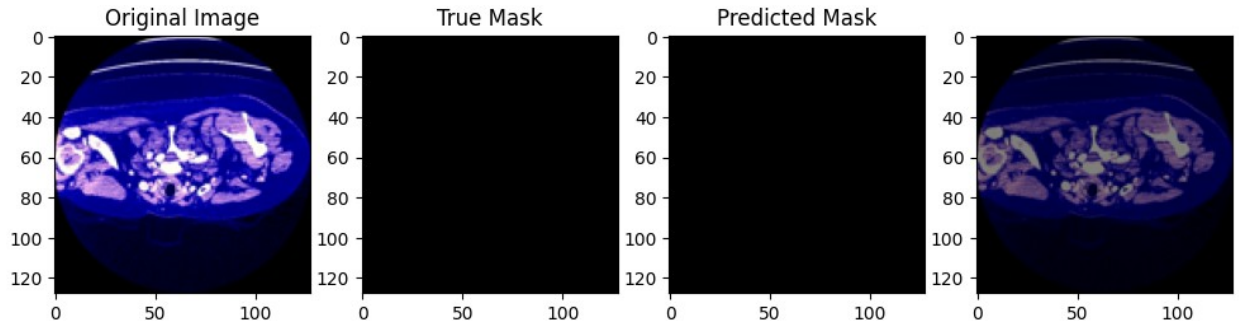
```
900
```

```
import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 50, replace=False)


for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model2
    predicted_mask = model2.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
    threshold = 0.5  # Adjust this threshold based on your model's
```

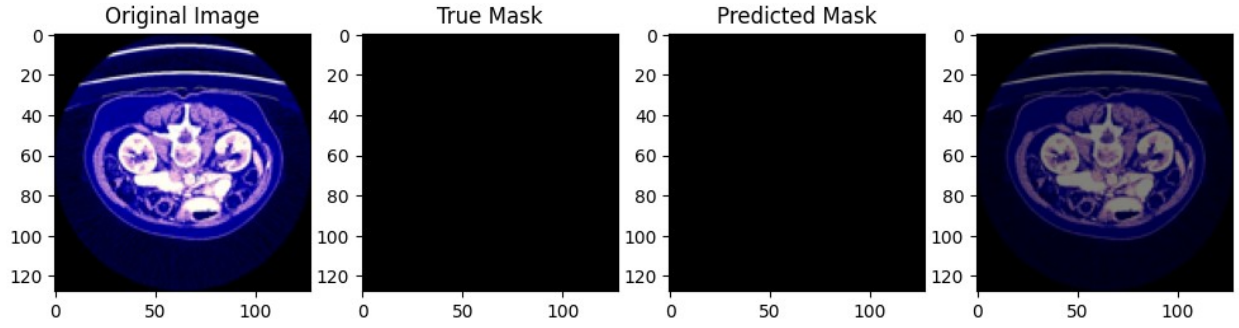
```
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')

    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
'nipy_spectral')

    plt.show()

333
1/1 [==============================] - 0s 189ms/step
```



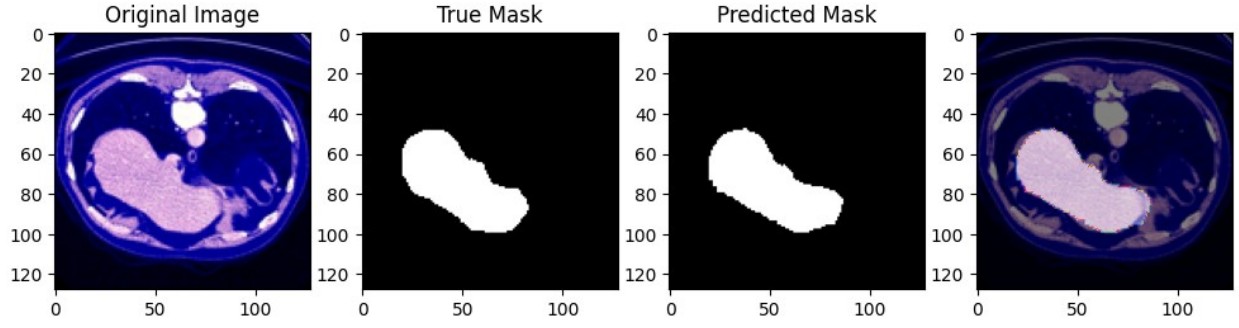

```
812
1/1 [==============================] - 0s 22ms/step
```



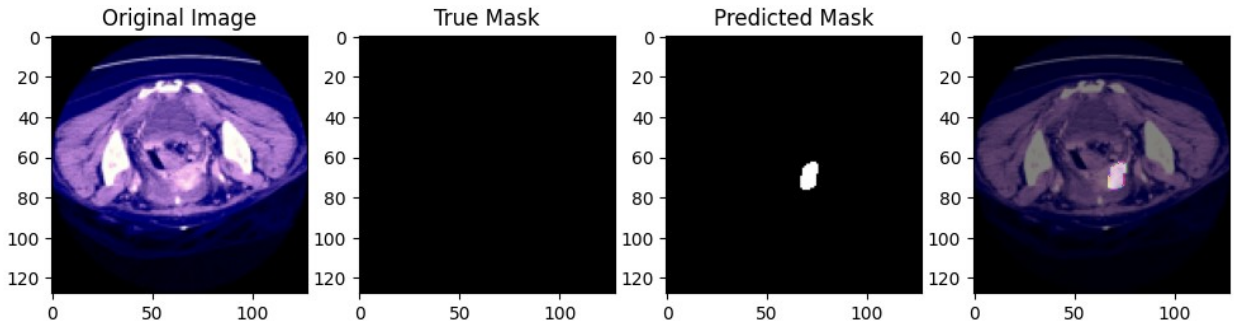

```
745
1/1 [==============================] - 0s 23ms/step
```



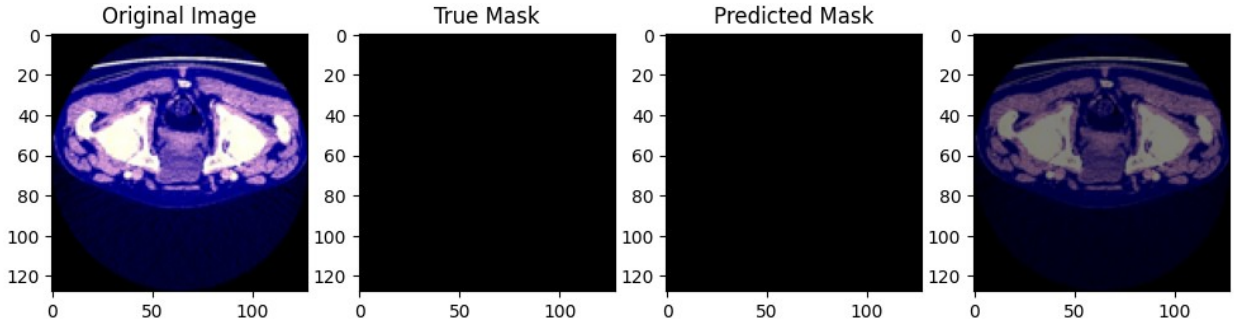

```
245
1/1 [==============================] - 0s 24ms/step
```



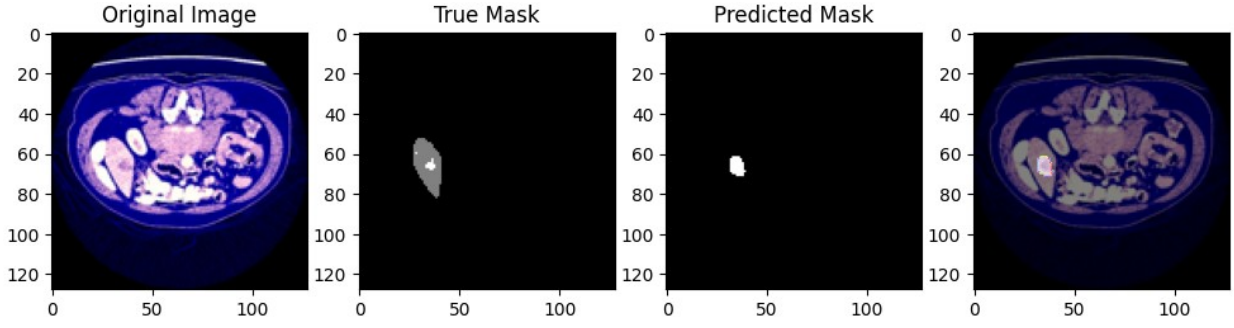

```
40
1/1 [==============================] - 0s 21ms/step
```



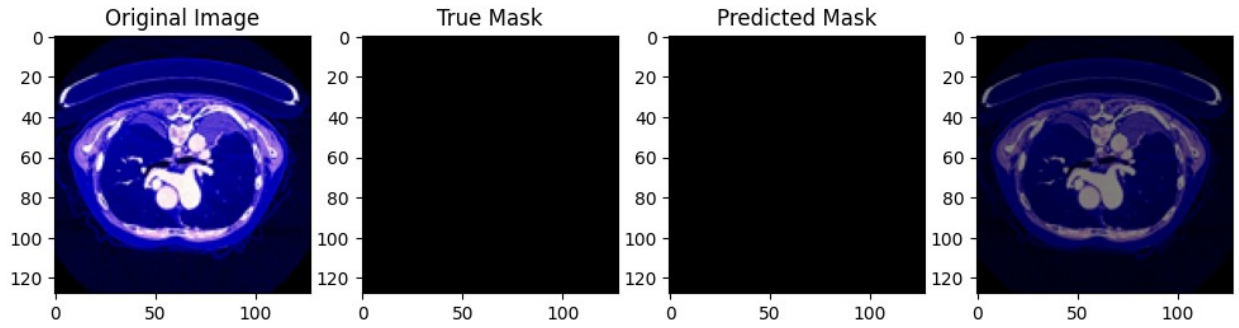

```
427
1/1 [==============================] - 0s 22ms/step
```



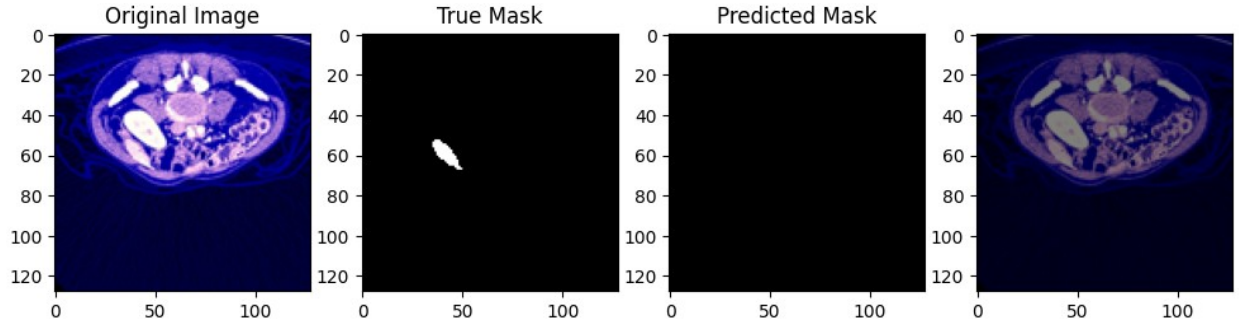

```
303
1/1 [==============================] - 0s 23ms/step
```



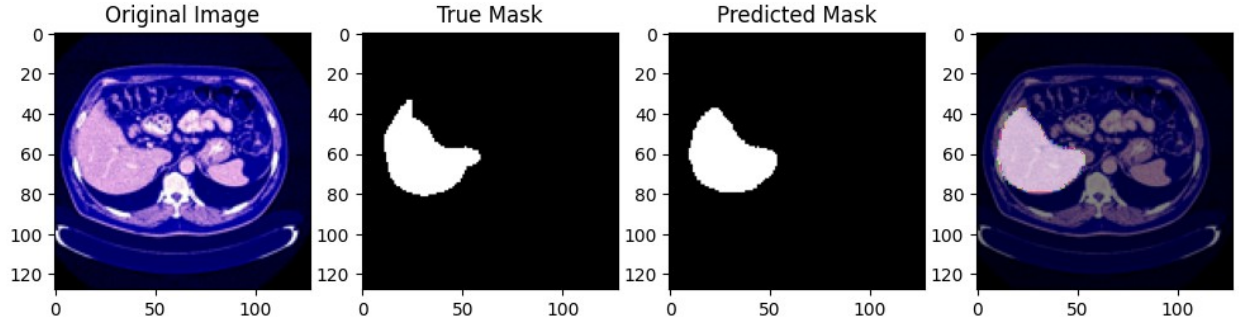

```
588
1/1 [==============================] - 0s 26ms/step
```



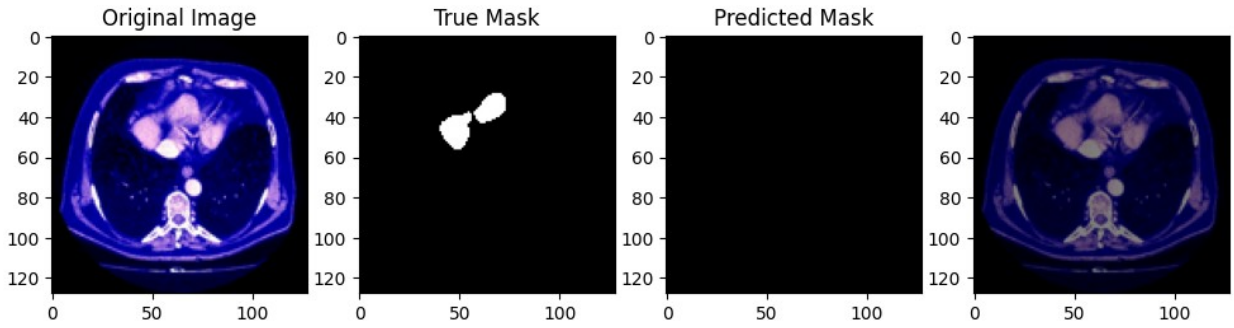

```
210
1/1 [==============================] - 0s 23ms/step
```



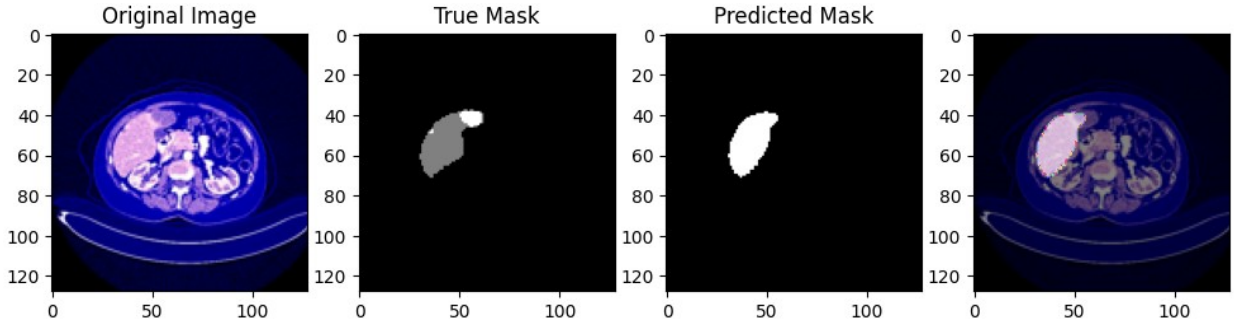

```
137
1/1 [==============================] - 0s 22ms/step
```



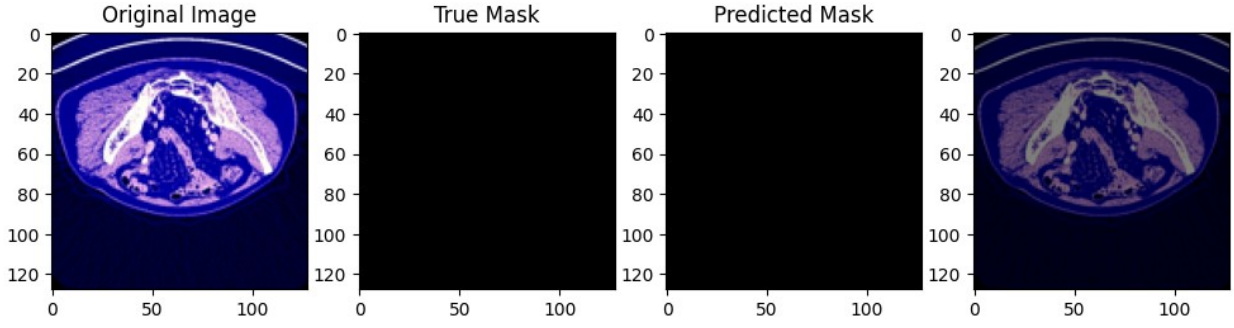

```
138
1/1 [==============================] - 0s 23ms/step
```



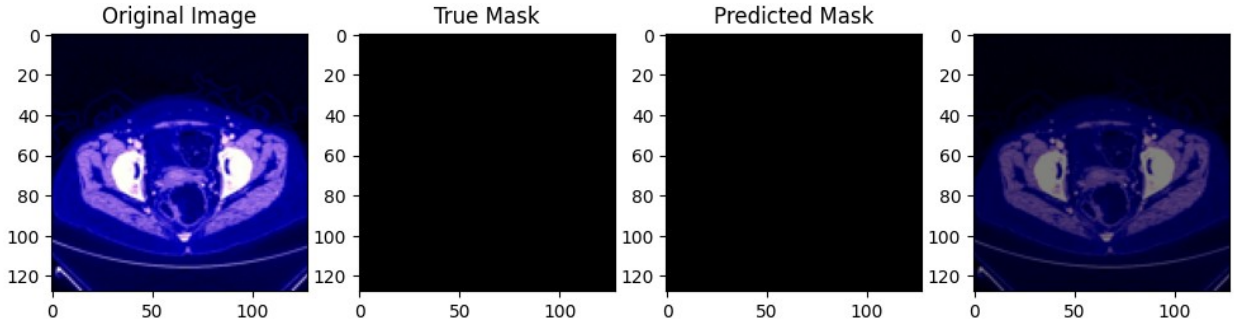

```
261
1/1 [==============================] - 0s 22ms/step
```



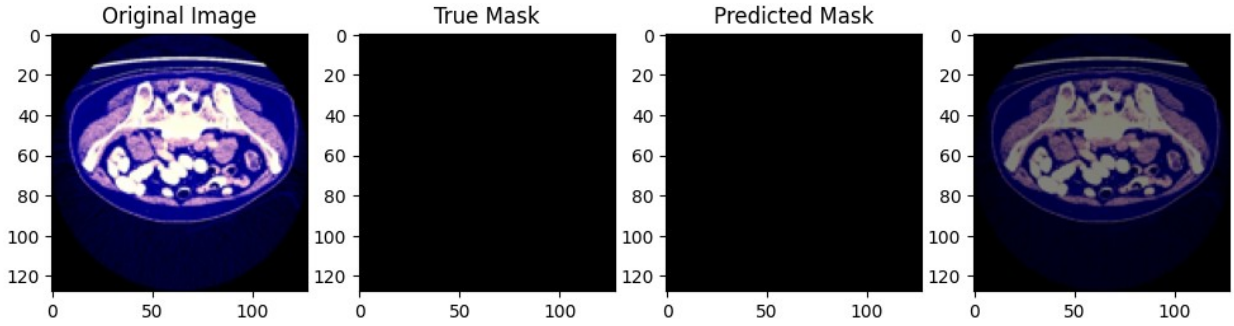

```
352
1/1 [==============================] - 0s 23ms/step
```



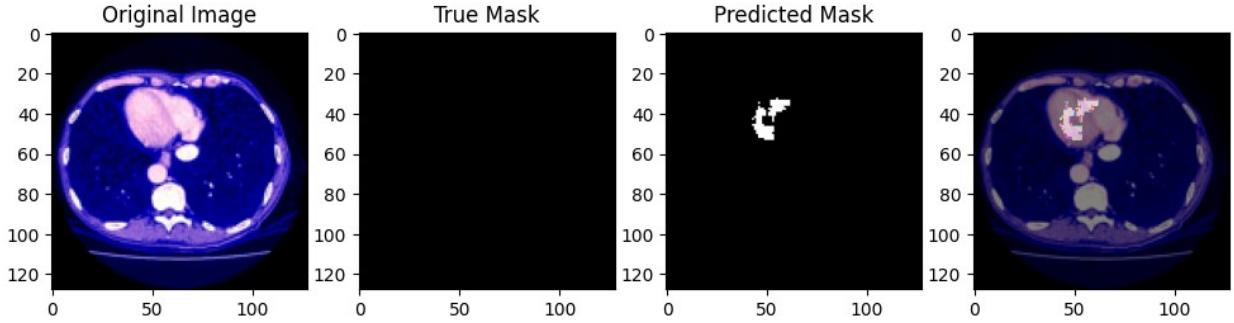

```
360
1/1 [==============================] - 0s 22ms/step
```



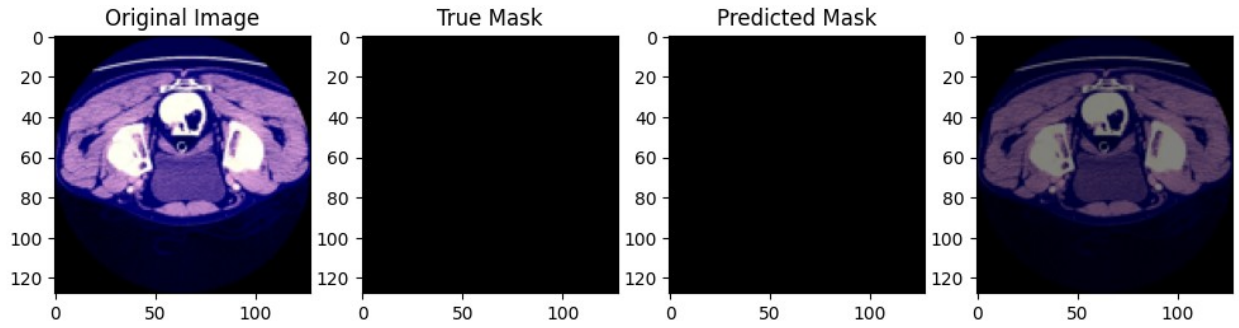

```
347
1/1 [==============================] - 0s 22ms/step
```



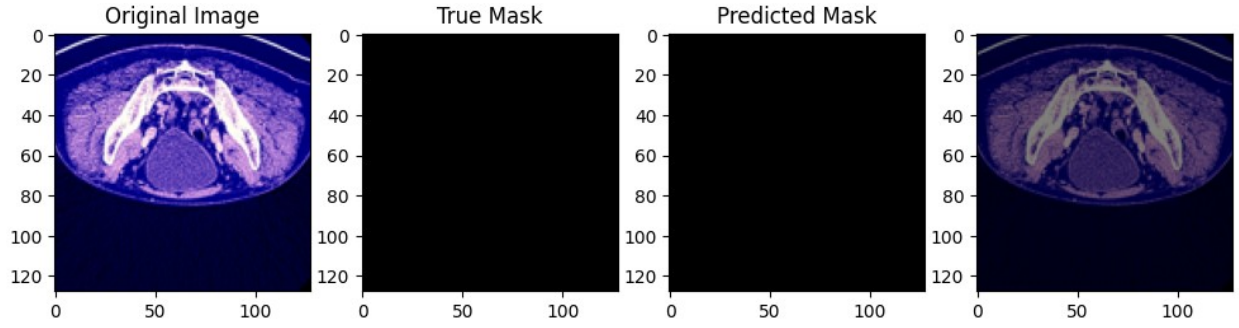

```
479
1/1 [==============================] - 0s 21ms/step
```



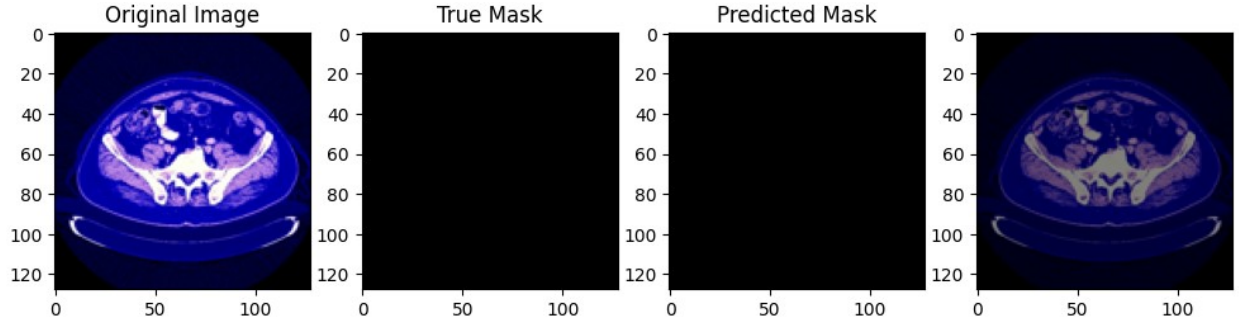

```
307
1/1 [==============================] - 0s 25ms/step
```



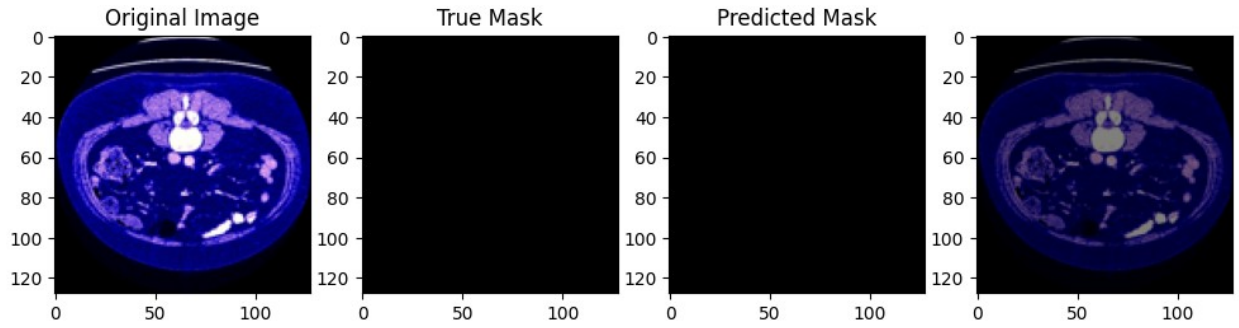

```
658
1/1 [==============================] - 0s 22ms/step
```



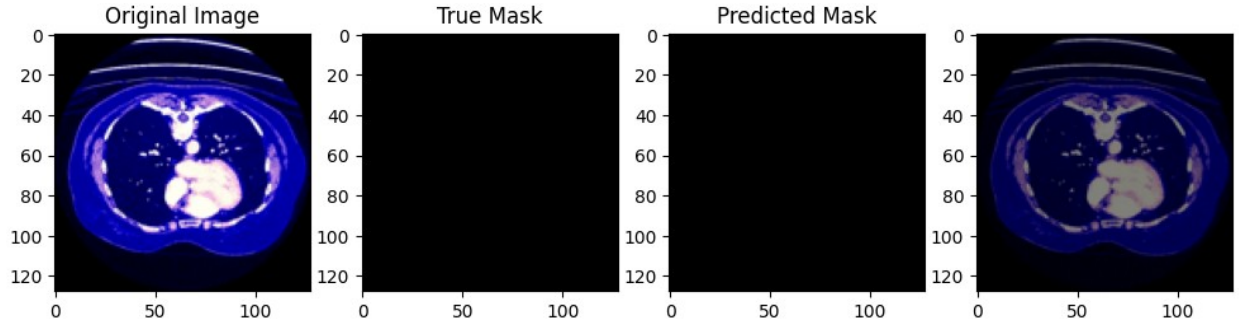

```
895
1/1 [==============================] - 0s 22ms/step
```



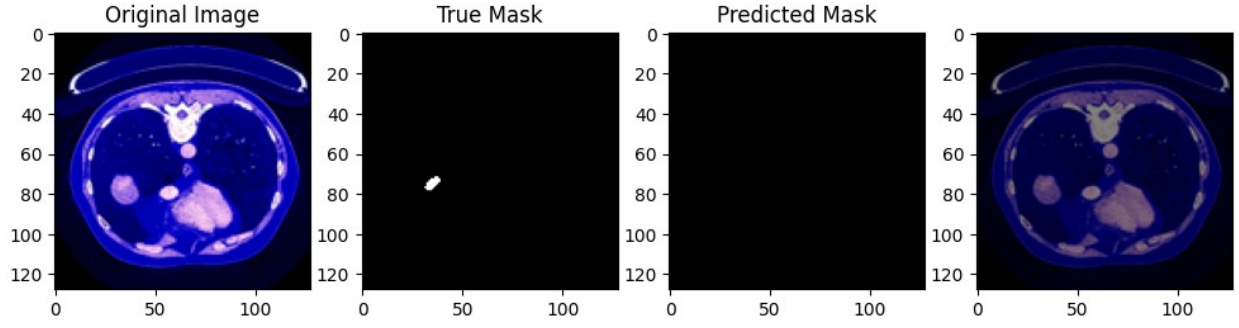

```
111
1/1 [==============================] - 0s 22ms/step
```



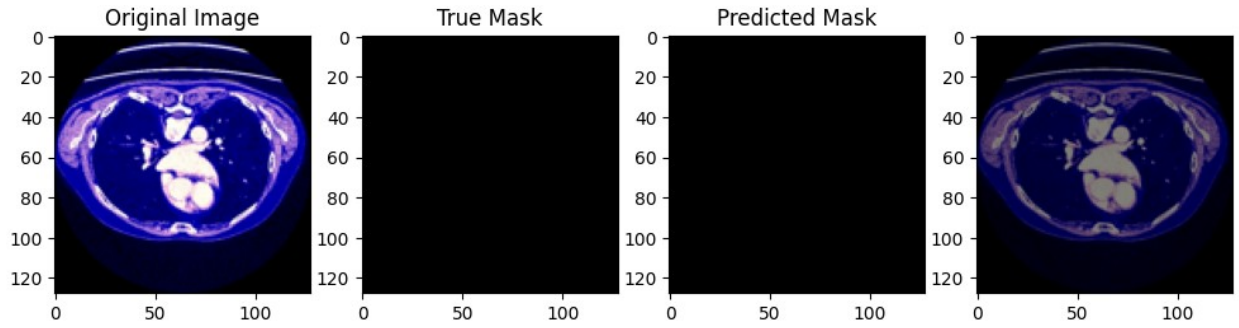

```
297
1/1 [==============================] - 0s 22ms/step
```



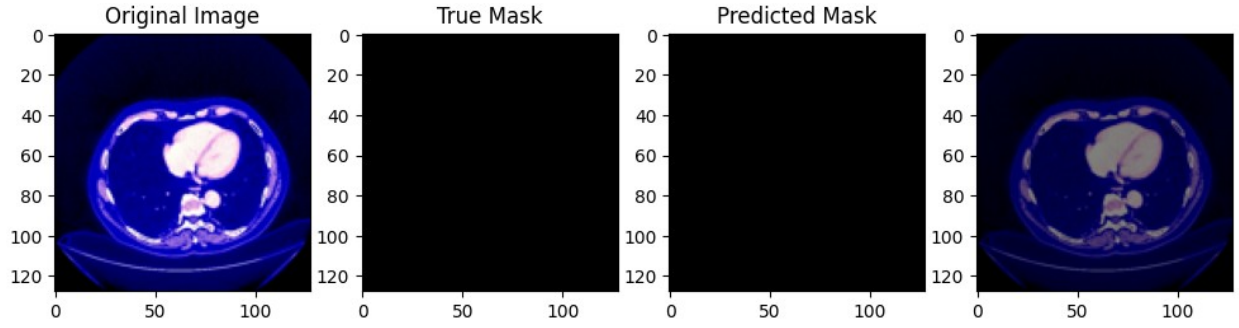

```
452
1/1 [==============================] - 0s 22ms/step
```



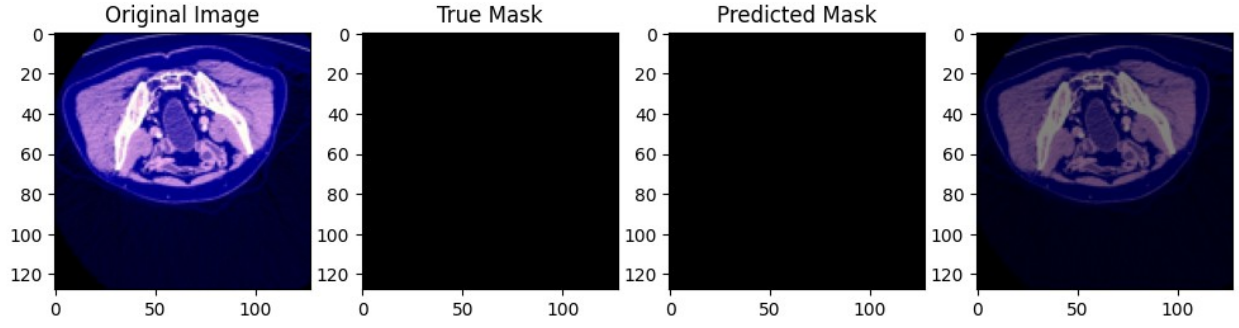

```
193
1/1 [==============================] - 0s 22ms/step
```



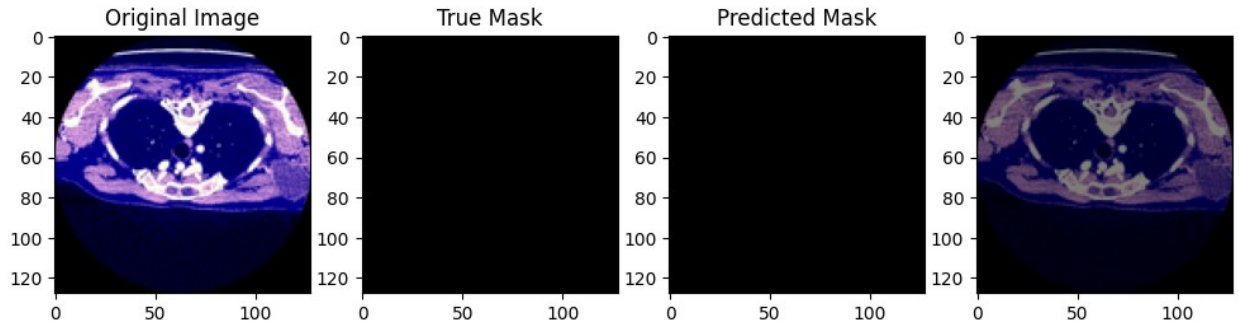

```
863
1/1 [==============================] - 0s 22ms/step
```



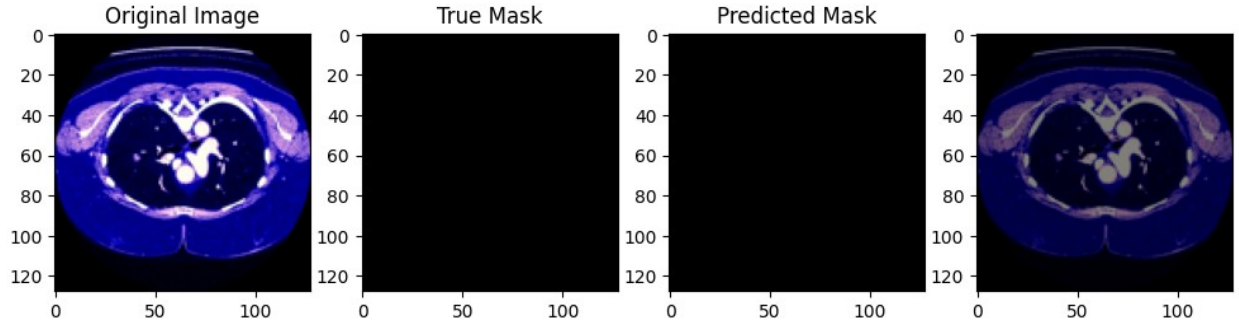

```
542
1/1 [==============================] - 0s 26ms/step
```



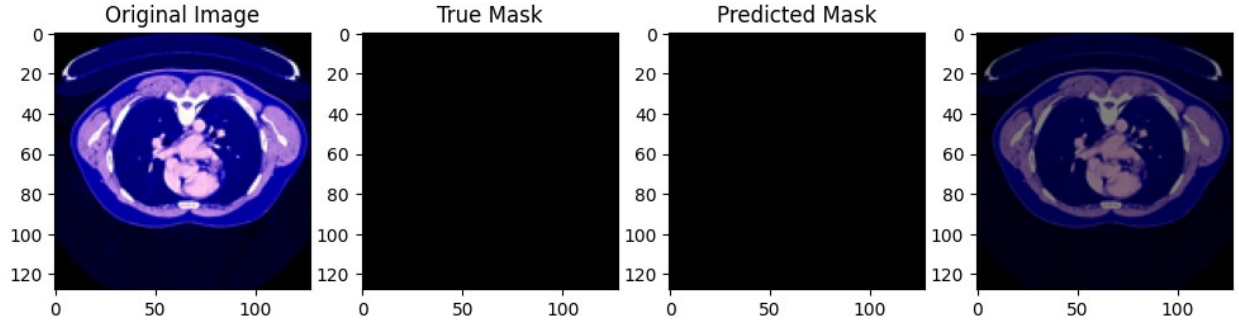

```
855
1/1 [==============================] - 0s 23ms/step
```



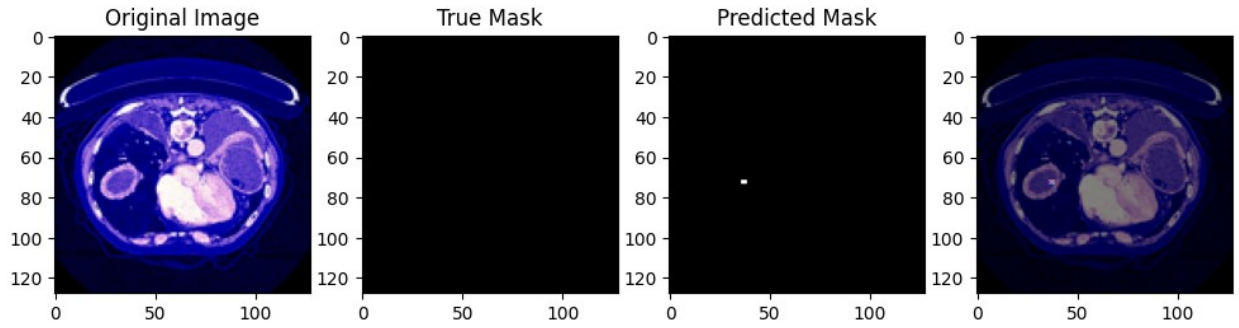

```
31
1/1 [==============================] - 0s 23ms/step
```



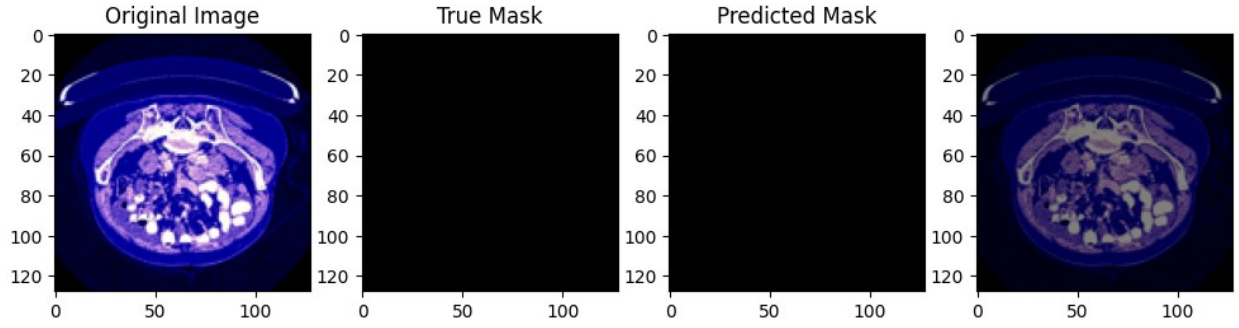

```
818
1/1 [==============================] - 0s 22ms/step
```



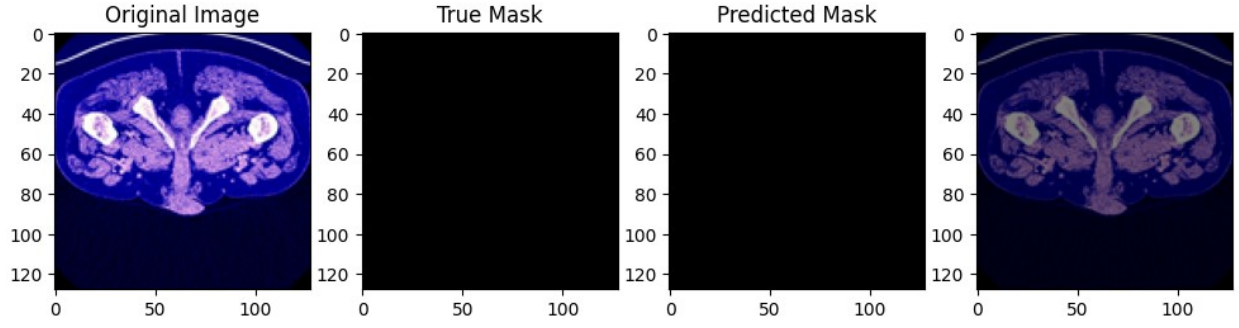

```
64
1/1 [==============================] - 0s 23ms/step
```



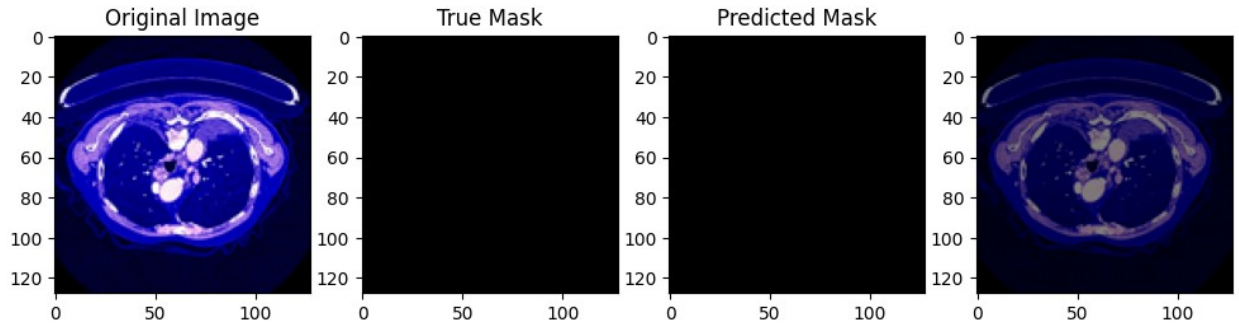

```
709
1/1 [==============================] - 0s 23ms/step
```



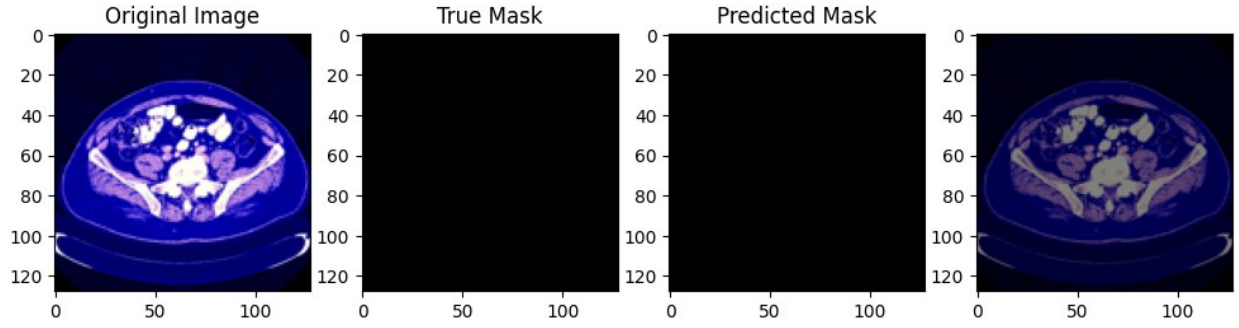

```
260
1/1 [==============================] - 0s 24ms/step
```



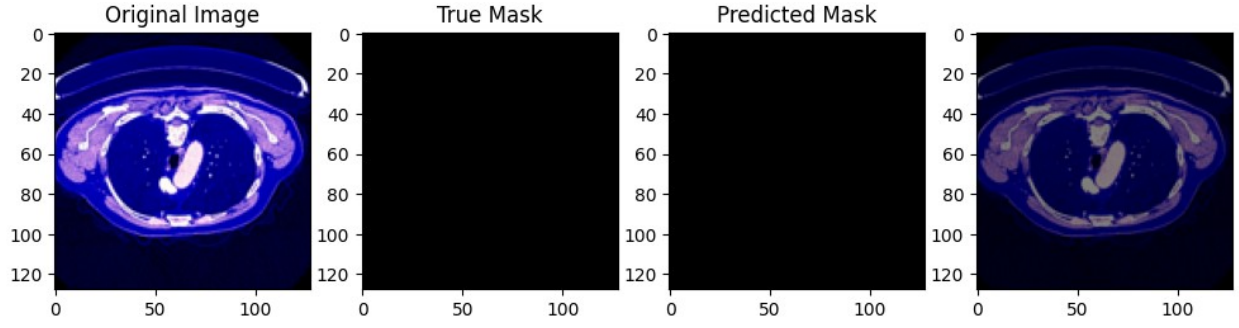

```
142
1/1 [==============================] - 0s 22ms/step
```



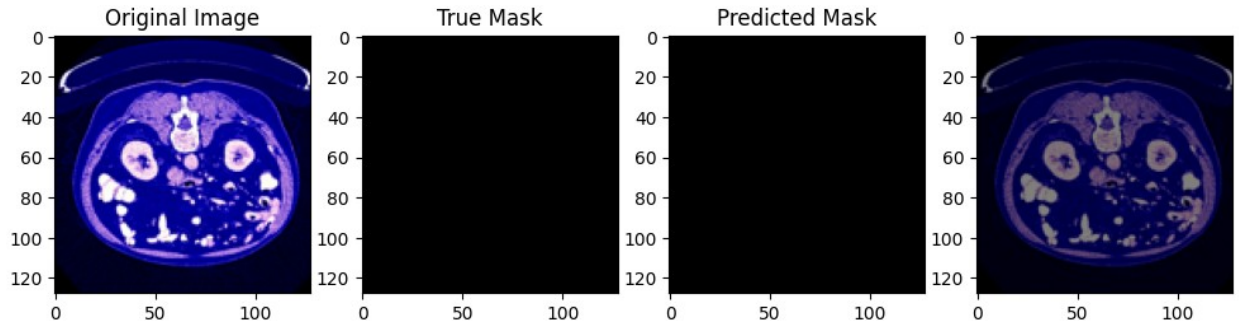

```
631
1/1 [==============================] - 0s 24ms/step
```



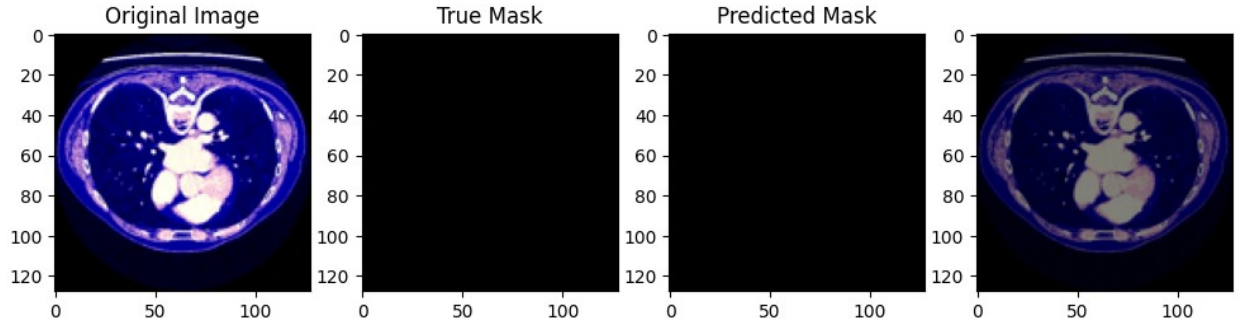

```
430
1/1 [==============================] - 0s 24ms/step
```



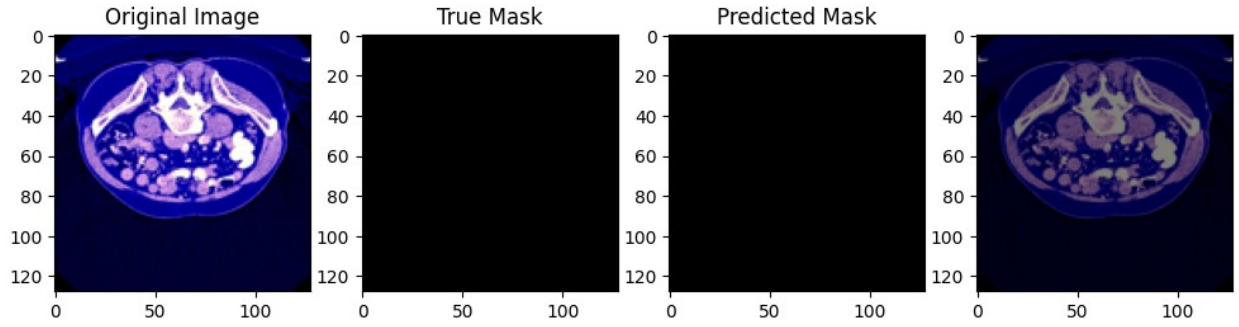

```
328
1/1 [==============================] - 0s 22ms/step
```



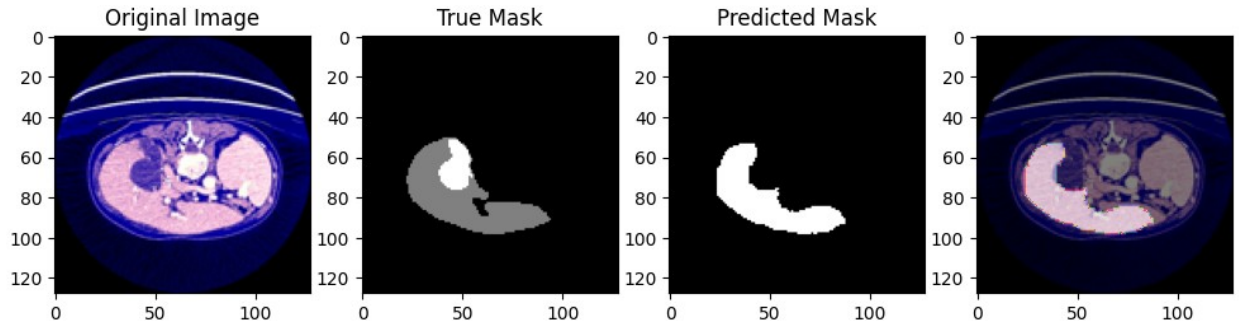

```
626
1/1 [==============================] - 0s 25ms/step
```



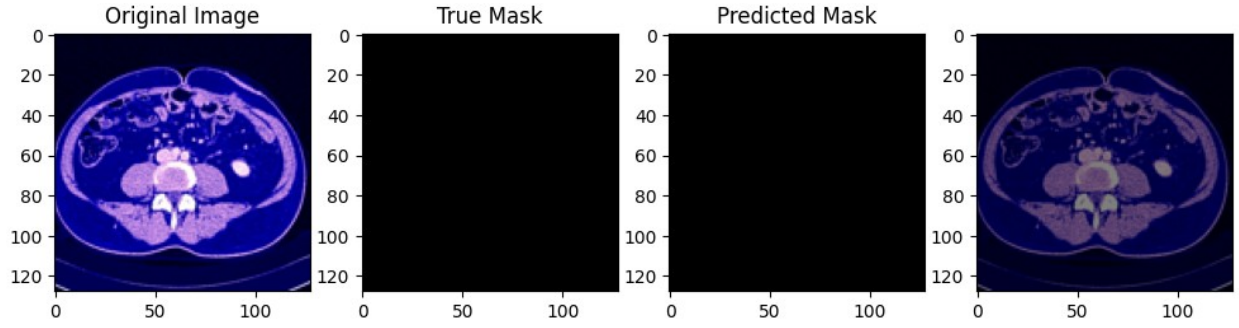

```
24
1/1 [==============================] - 0s 22ms/step
```



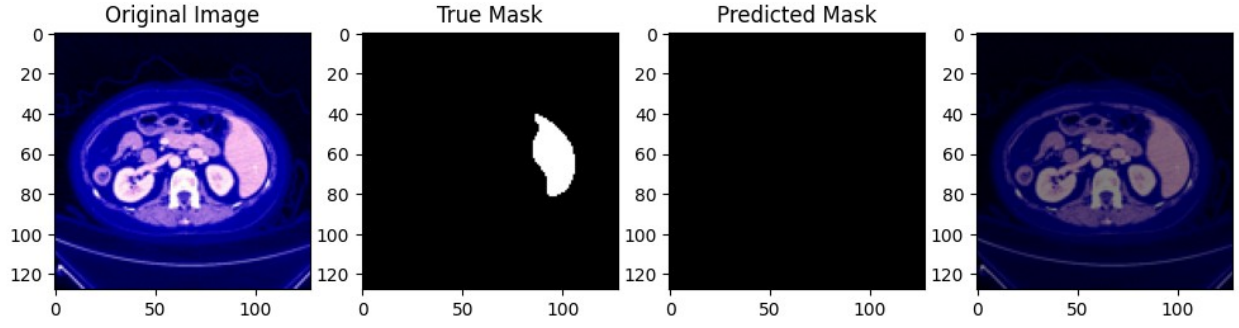

```
199
1/1 [==============================] - 0s 23ms/step
```



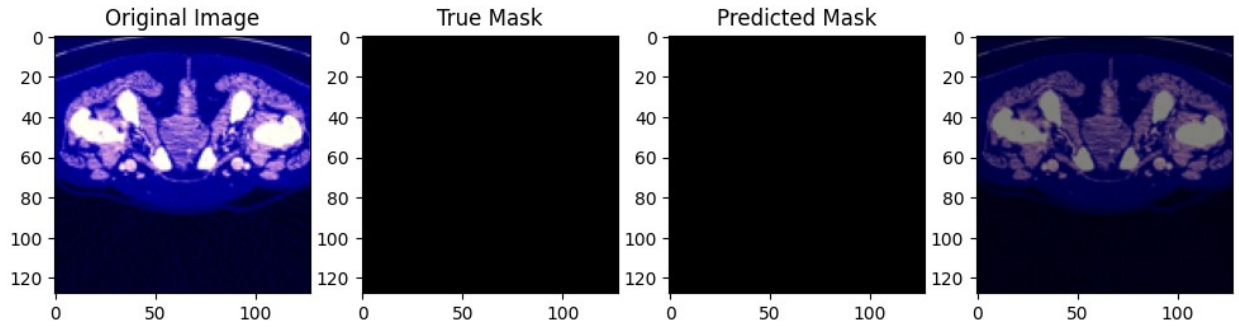

```
879
1/1 [==============================] - 0s 23ms/step
```



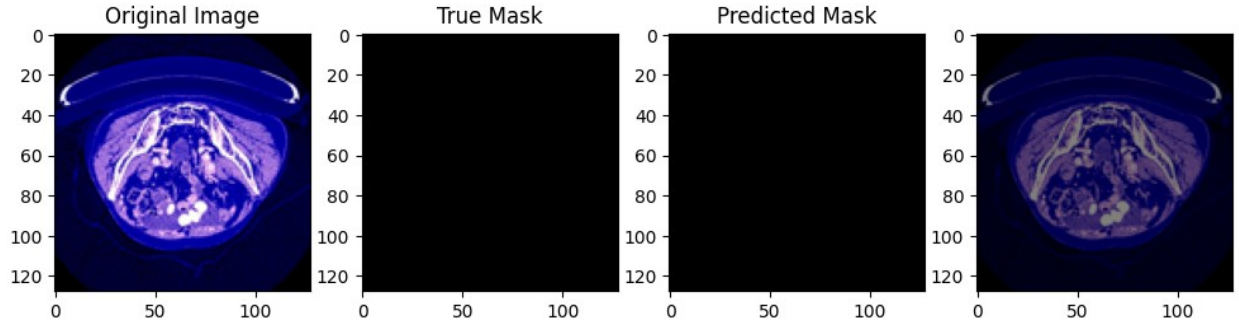

```
891
1/1 [==============================] - 0s 23ms/step
```



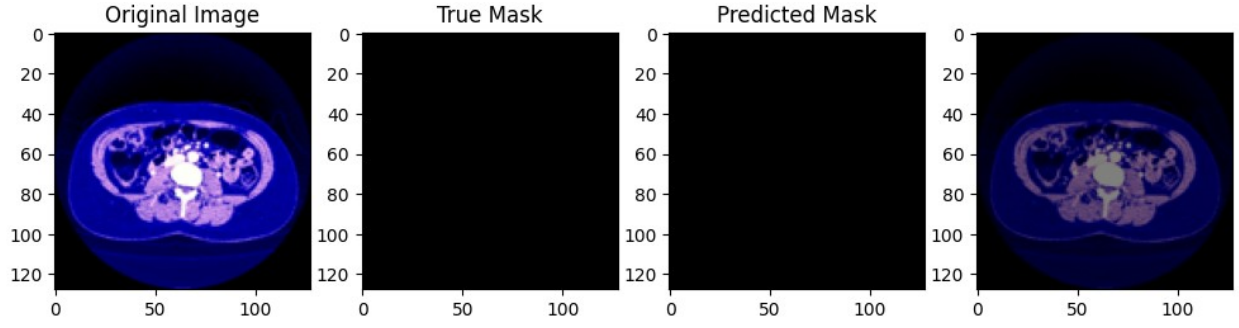

```
121
1/1 [==============================] - 0s 25ms/step
```



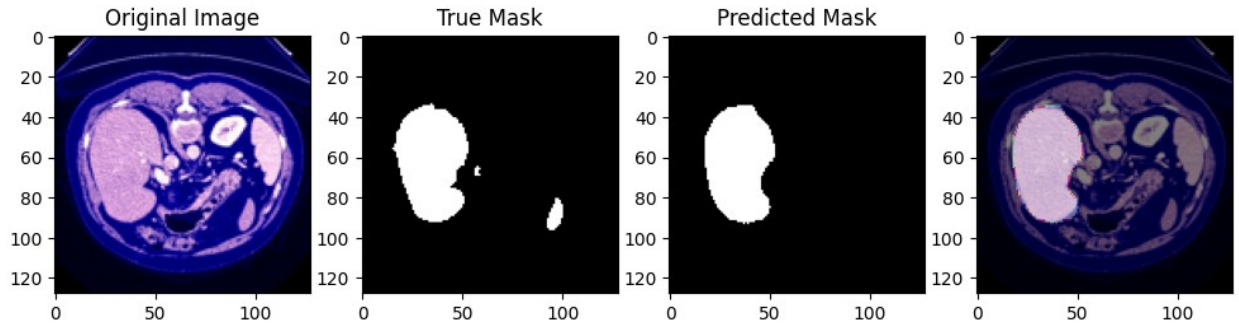

```
339
1/1 [==============================] - 0s 25ms/step
```



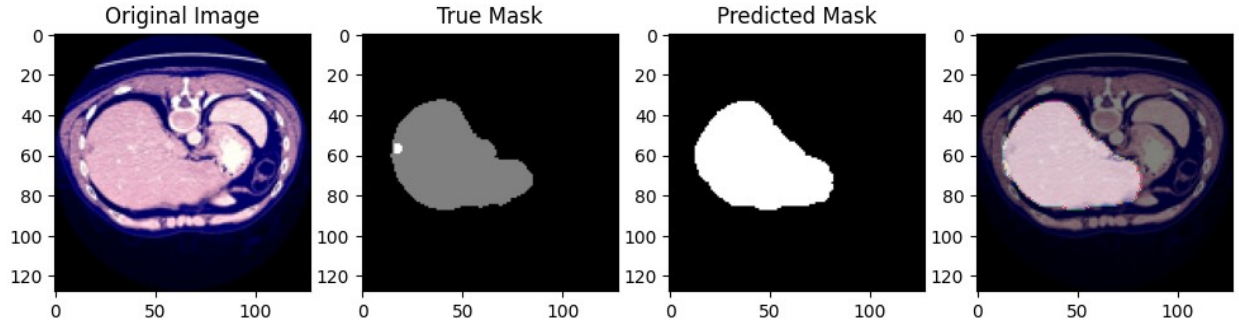

```
652
1/1 [==============================] - 0s 25ms/step
```



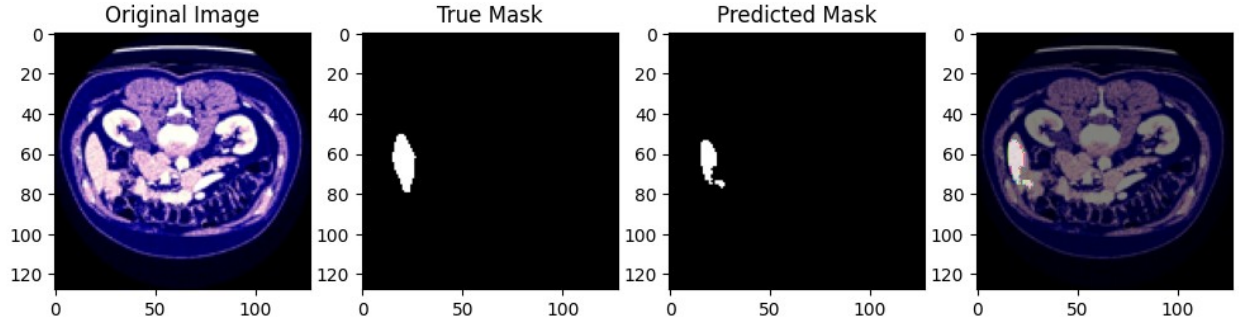

```
744
1/1 [==============================] - 0s 22ms/step
```



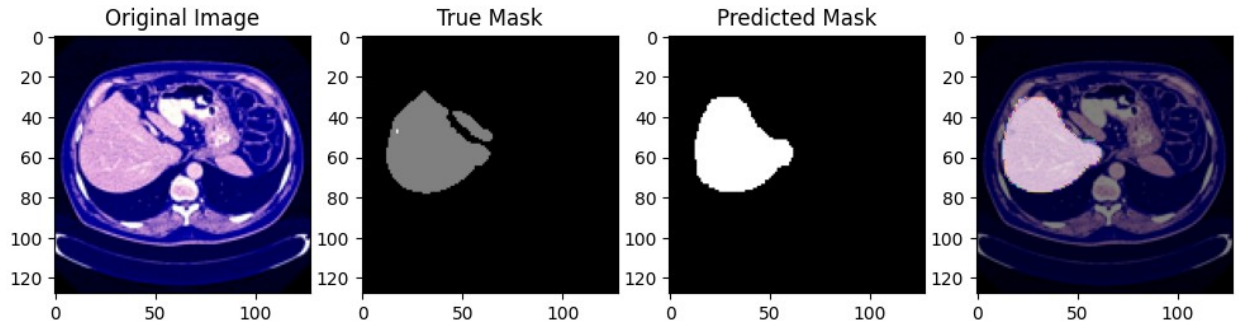

```
366
1/1 [==============================] - 0s 26ms/step
```



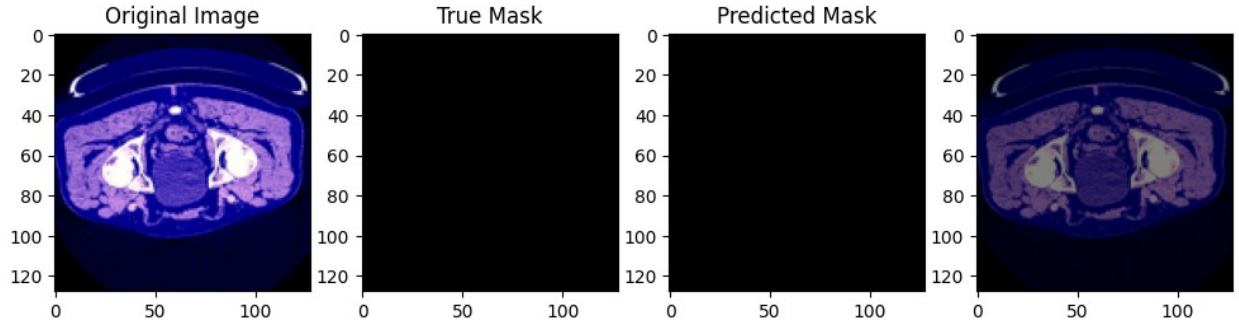

```
87
1/1 [==============================] - 0s 23ms/step
```



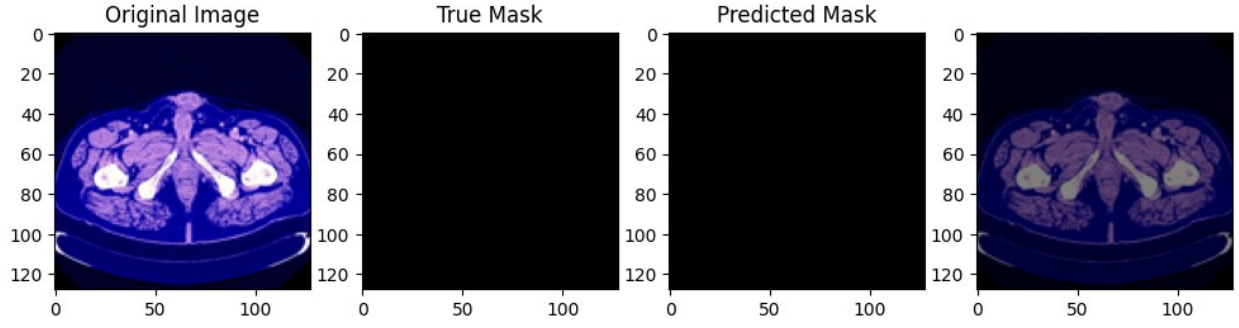

```
68
1/1 [==============================] - 0s 22ms/step
```



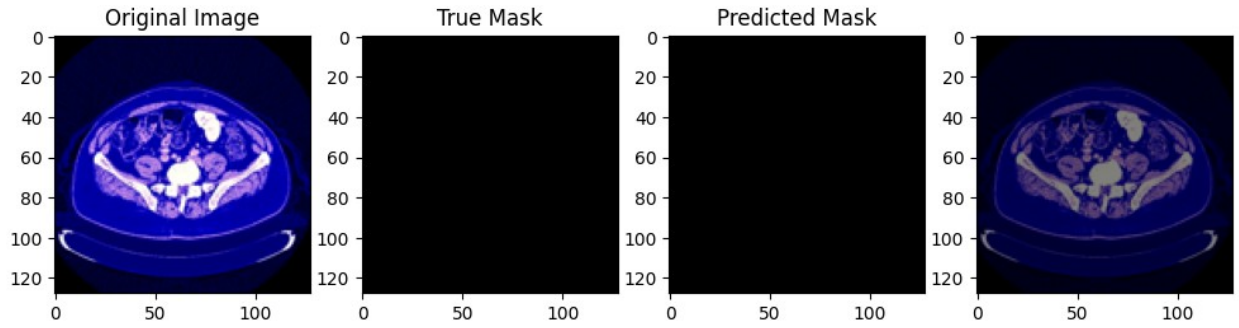

```
353
1/1 [==============================] - 0s 24ms/step
```




```
212
1/1 [==============================] - 0s 22ms/step
```




```
741
1/1 [==============================] - 0s 24ms/step
```

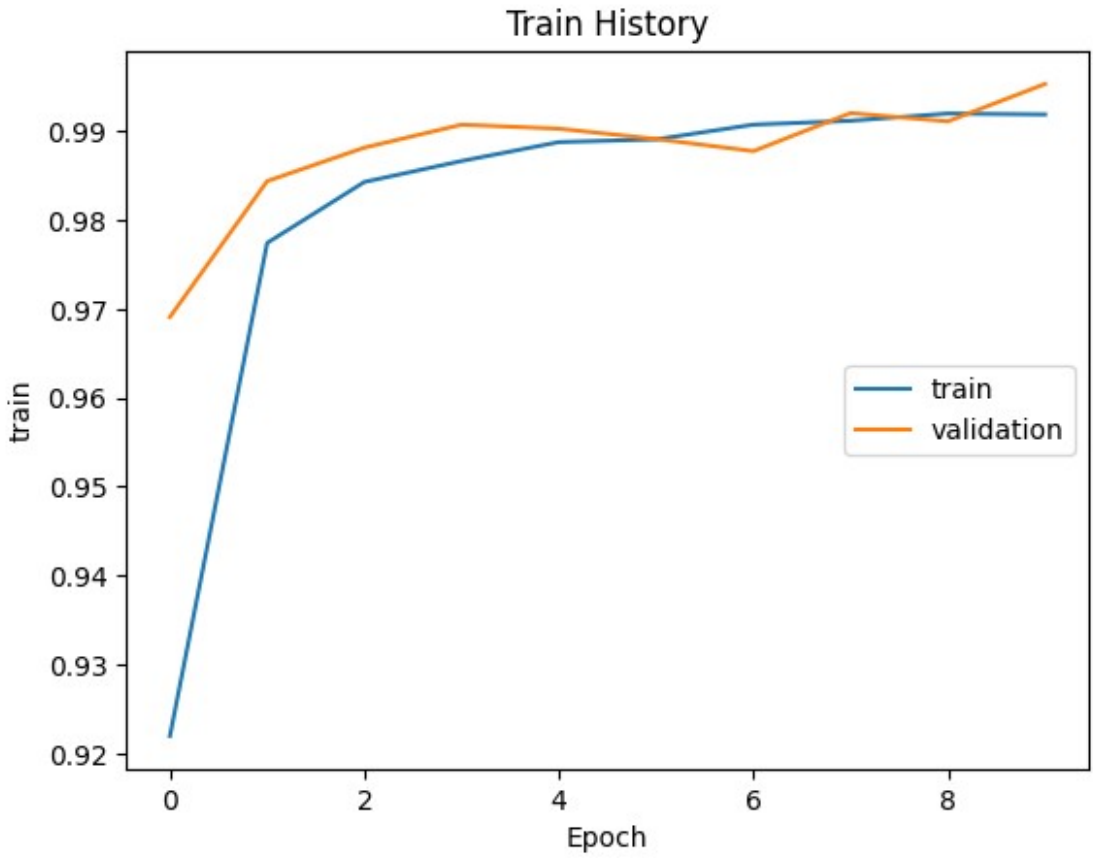



```
# weights to the predictions
unet_weight = 0.98
fcn_weight = 0.71
segnet_weight = 0.99
sum_weights = unet_weight + fcn_weight + segnet_weight

w_unet = unet_weight / sum_weights
w_fcn = fcn_weight / sum_weights
w_segnet = segnet_weight / sum_weights

#element-wise maximum prediction
weighted_prediction = np.maximum(w_unet * prediction1, w_fcn *
prediction, w_segnet * prediction2)

#argmax along the last axis
final_prediction = np.argmax(weighted_prediction, axis=-1)
# testing score using the combined predictions
test_scores1 = model2.evaluate(x_test, y_test)
test_scores2 = model1.evaluate(x_test, y_test)
weighted_test_score = w_unet * test_scores[1] + w_fcn *
model.evaluate(x_test, y_test)[1] + w_segnet*test_scores1[1]

print(f'Testing score with weighted predictions:
{weighted_test_score}')
```

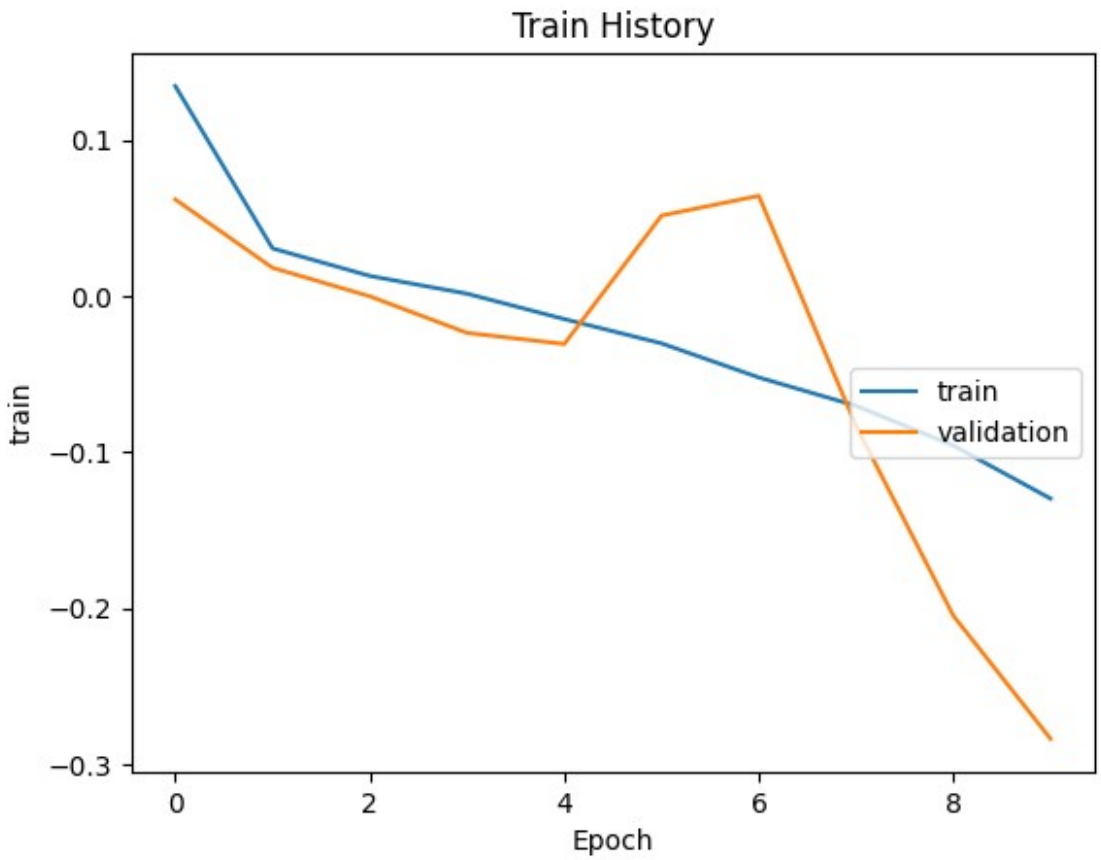
```
29/29 [==============================] - 2s 54ms/step - loss: -0.3152
- dice_coefficient: 0.9953
29/29 [==============================] - 1s 18ms/step - loss: 0.0253 -
dice_coefficient: 0.9878
29/29 [==============================] - 1s 25ms/step - loss: 0.0355 -
dice_coefficient: 0.6273
Testing score with weighted predictions: 0.7632188770308423
```

```
import matplotlib.pyplot as plt
def show_history(history, train, validation,x=1):
    plt.plot(history.history[train])
    plt.plot(history.history[validation])
    plt.title('Train History')
    plt.ylabel('train')
    plt.xlabel('Epoch')
```

```
    plt.legend(['train', 'validation'], loc='center right')
    plt.savefig(f'{train}dice_coefficient_{str(x)}.png')
    plt.show()
show_history(history2, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history2, 'loss', 'val_loss')
```




```
def bn_act(x, act=True):
    x = keras.layers.BatchNormalization()(x)
    if act == True:
        x = keras.layers.Activation("relu")(x)
    return x
import tensorflow as tf
from tensorflow import keras
def conv_block(x, filters, kernel_size=(3, 3), padding="same",
strides=1):
    conv = bn_act(x)
    conv = keras.layers.Conv2D(filters, kernel_size, padding=padding,
strides=strides)(conv)
    return conv

def stem(x, filters, kernel_size=(3, 3), padding="same", strides=1):
    conv = keras.layers.Conv2D(filters, kernel_size, padding=padding,
strides=strides)(x)
    conv = conv_block(conv, filters, kernel_size=kernel_size,
padding=padding, strides=strides)

    shortcut = keras.layers.Conv2D(filters, kernel_size=(1, 1),
padding=padding, strides=strides)(x)
    shortcut = bn_act(shortcut, act=False)
```

```
    output = keras.layers.Add()([conv, shortcut])
    return output

def residual_block(x, filters, kernel_size=(3, 3), padding="same",
strides=1):
    res = conv_block(x, filters, kernel_size=kernel_size,
padding=padding, strides=strides)
    res = conv_block(res, filters, kernel_size=kernel_size,
padding=padding, strides=1)

    shortcut = keras.layers.Conv2D(filters, kernel_size=(1, 1),
padding=padding, strides=strides)(x)
    shortcut = bn_act(shortcut, act=False)

    output = keras.layers.Add()([shortcut, res])
    return output

def upsample_concat_block(x, xskip):
    u = keras.layers.UpSampling2D((2, 2))(x)
    c = keras.layers.Concatenate()([u, xskip])
    return c

def ResUNet():
    f = [32, 64, 128, 256, 512]  # Adjusted filter sizes for deeper
network
    inputs = keras.layers.Input((128, 128, 3))

    # Encoder
    e0 = inputs
    e1 = stem(e0, f[0])
    e2 = residual_block(e1, f[1], strides=2)
    e3 = residual_block(e2, f[2], strides=2)
    e4 = residual_block(e3, f[3], strides=2)
    e5 = residual_block(e4, f[4], strides=2)

    # Bridge
    b0 = conv_block(e5, f[4], strides=1)
    b1 = conv_block(b0, f[4], strides=1)

    # Decoder
    u1 = upsample_concat_block(b1, e4)
    d1 = residual_block(u1, f[3])

    u2 = upsample_concat_block(d1, e3)
    d2 = residual_block(u2, f[2])

    u3 = upsample_concat_block(d2, e2)
    d3 = residual_block(u3, f[1])
```

```
    u4 = upsample_concat_block(d3, e1)
    d4 = residual_block(u4, f[0])

    # Output
    outputs = keras.layers.Conv2D(1, (1, 1), padding="same",
activation="sigmoid")(d4)
    model = keras.models.Model(inputs, outputs)
    return model

# Instantiate the model
model3 = ResUNet()

# Compile the model
model3.compile(optimizer='Adam', loss='categorical_crossentropy',
metrics=[dice_coefficient])

# Display the model summary
model3.summary()
```

```
Model: "model_1"
__________________________________________________________________
_______________________________
 Layer (type)                Output Shape              Param #
Connected to
==================================================================
===============================
 input_2 (InputLayer)        [(None, 128, 128, 3)]     0         []


 conv2d_36 (Conv2D)          (None, 128, 128, 32)      896
['input_2[0][0]']


 batch_normalization_23 (Ba  (None, 128, 128, 32)      128
['conv2d_36[0][0]']
 tchNormalization)


 activation_14 (Activation)  (None, 128, 128, 32)      0
['batch_normalization_23[0][0]
                                                                  ']


 conv2d_38 (Conv2D)          (None, 128, 128, 32)      128
['input_2[0][0]']
```

```
 conv2d_37 (Conv2D)          (None, 128, 128, 32)         9248
['activation_14[0][0]']


 batch_normalization_24 (Ba  (None, 128, 128, 32)         128
['conv2d_38[0][0]']
 tchNormalization)



 add (Add)                   (None, 128, 128, 32)         0
['conv2d_37[0][0]',

'batch_normalization_24[0][0]
                                                                  ']



 batch_normalization_25 (Ba  (None, 128, 128, 32)         128
['add[0][0]']
 tchNormalization)



 activation_15 (Activation)  (None, 128, 128, 32)         0
['batch_normalization_25[0][0]
                                                                  ']



 conv2d_39 (Conv2D)          (None, 64, 64, 64)           18496
['activation_15[0][0]']


 batch_normalization_26 (Ba  (None, 64, 64, 64)           256
['conv2d_39[0][0]']
 tchNormalization)



 conv2d_41 (Conv2D)          (None, 64, 64, 64)           2112
['add[0][0]']


 activation_16 (Activation)  (None, 64, 64, 64)           0
['batch_normalization_26[0][0]
                                                                  ']



 batch_normalization_27 (Ba  (None, 64, 64, 64)           256
```

```
['conv2d_41[0][0]']
 tchNormalization)



 conv2d_40 (Conv2D)          (None, 64, 64, 64)           36928
['activation_16[0][0]']


 add_1 (Add)                 (None, 64, 64, 64)           0
['batch_normalization_27[0][0]
                                                                 ',


'conv2d_40[0][0]']


 batch_normalization_28 (Ba  (None, 64, 64, 64)           256
['add_1[0][0]']
 tchNormalization)



 activation_17 (Activation)  (None, 64, 64, 64)           0
['batch_normalization_28[0][0]
                                                                 ']



 conv2d_42 (Conv2D)          (None, 32, 32, 128)          73856
['activation_17[0][0]']


 batch_normalization_29 (Ba  (None, 32, 32, 128)          512
['conv2d_42[0][0]']
 tchNormalization)



 conv2d_44 (Conv2D)          (None, 32, 32, 128)          8320
['add_1[0][0]']


 activation_18 (Activation)  (None, 32, 32, 128)          0
['batch_normalization_29[0][0]
                                                                 ']



 batch_normalization_30 (Ba  (None, 32, 32, 128)          512
['conv2d_44[0][0]']
```

```
 tchNormalization)


 conv2d_43 (Conv2D)          (None, 32, 32, 128)          147584
['activation_18[0][0]']

 add_2 (Add)                 (None, 32, 32, 128)          0
['batch_normalization_30[0][0]
                                                                  ',

'conv2d_43[0][0]']

 batch_normalization_31 (Ba  (None, 32, 32, 128)          512
['add_2[0][0]']
 tchNormalization)

 activation_19 (Activation)  (None, 32, 32, 128)          0
['batch_normalization_31[0][0]
                                                                  ']

 conv2d_45 (Conv2D)          (None, 16, 16, 256)          295168
['activation_19[0][0]']

 batch_normalization_32 (Ba  (None, 16, 16, 256)          1024
['conv2d_45[0][0]']
 tchNormalization)

 conv2d_47 (Conv2D)          (None, 16, 16, 256)          33024
['add_2[0][0]']

 activation_20 (Activation)  (None, 16, 16, 256)          0
['batch_normalization_32[0][0]
                                                                  ']

 batch_normalization_33 (Ba  (None, 16, 16, 256)          1024
['conv2d_47[0][0]']
 tchNormalization)
```

```
 conv2d_46 (Conv2D)          (None, 16, 16, 256)          590080
['activation_20[0][0]']

 add_3 (Add)                 (None, 16, 16, 256)          0
['batch_normalization_33[0][0]
                                                                  ',

'conv2d_46[0][0]']

 batch_normalization_34 (Ba  (None, 16, 16, 256)          1024
['add_3[0][0]']
 tchNormalization)

 activation_21 (Activation)  (None, 16, 16, 256)          0
['batch_normalization_34[0][0]
                                                                  ']

 conv2d_48 (Conv2D)          (None, 8, 8, 512)            1180160
['activation_21[0][0]']

 batch_normalization_35 (Ba  (None, 8, 8, 512)            2048
['conv2d_48[0][0]']
 tchNormalization)

 conv2d_50 (Conv2D)          (None, 8, 8, 512)            131584
['add_3[0][0]']

 activation_22 (Activation)  (None, 8, 8, 512)            0
['batch_normalization_35[0][0]
                                                                  ']

 batch_normalization_36 (Ba  (None, 8, 8, 512)            2048
['conv2d_50[0][0]']
 tchNormalization)
```

```
 conv2d_49 (Conv2D)          (None, 8, 8, 512)            2359808
['activation_22[0][0]']


 add_4 (Add)                 (None, 8, 8, 512)            0
['batch_normalization_36[0][0]
                                                                  ',


'conv2d_49[0][0]']


 batch_normalization_37 (Ba  (None, 8, 8, 512)            2048
['add_4[0][0]']
 tchNormalization)



 activation_23 (Activation)  (None, 8, 8, 512)            0
['batch_normalization_37[0][0]
                                                                  ']



 conv2d_51 (Conv2D)          (None, 8, 8, 512)            2359808
['activation_23[0][0]']


 batch_normalization_38 (Ba  (None, 8, 8, 512)            2048
['conv2d_51[0][0]']
 tchNormalization)



 activation_24 (Activation)  (None, 8, 8, 512)            0
['batch_normalization_38[0][0]
                                                                  ']



 conv2d_52 (Conv2D)          (None, 8, 8, 512)            2359808
['activation_24[0][0]']


 up_sampling2d_8 (UpSamplin  (None, 16, 16, 512)          0
['conv2d_52[0][0]']
 g2D)
```

```
 concatenate_2 (Concatenate  (None, 16, 16, 768)          0
['up_sampling2d_8[0][0]',
 )
'add_3[0][0]']


 batch_normalization_39 (Ba  (None, 16, 16, 768)          3072
['concatenate_2[0][0]']
 tchNormalization)



 activation_25 (Activation)  (None, 16, 16, 768)          0
['batch_normalization_39[0][0]
                                                                 ']



 conv2d_53 (Conv2D)          (None, 16, 16, 256)          1769728
['activation_25[0][0]']


 batch_normalization_40 (Ba  (None, 16, 16, 256)          1024
['conv2d_53[0][0]']
 tchNormalization)



 conv2d_55 (Conv2D)          (None, 16, 16, 256)          196864
['concatenate_2[0][0]']


 activation_26 (Activation)  (None, 16, 16, 256)          0
['batch_normalization_40[0][0]
                                                                 ']



 batch_normalization_41 (Ba  (None, 16, 16, 256)          1024
['conv2d_55[0][0]']
 tchNormalization)



 conv2d_54 (Conv2D)          (None, 16, 16, 256)          590080
['activation_26[0][0]']


 add_5 (Add)                 (None, 16, 16, 256)          0
['batch_normalization_41[0][0]
```

```
                                                                  ',
'conv2d_54[0][0]']

 up_sampling2d_9 (UpSamplin  (None, 32, 32, 256)          0
['add_5[0][0]']
 g2D)

 concatenate_3 (Concatenate  (None, 32, 32, 384)          0
['up_sampling2d_9[0][0]',
 )
'add_2[0][0]']

 batch_normalization_42 (Ba  (None, 32, 32, 384)          1536
['concatenate_3[0][0]']
 tchNormalization)

 activation_27 (Activation)  (None, 32, 32, 384)          0
['batch_normalization_42[0][0]
                                                                  ']

 conv2d_56 (Conv2D)          (None, 32, 32, 128)          442496
['activation_27[0][0]']

 batch_normalization_43 (Ba  (None, 32, 32, 128)          512
['conv2d_56[0][0]']
 tchNormalization)

 conv2d_58 (Conv2D)          (None, 32, 32, 128)          49280
['concatenate_3[0][0]']

 activation_28 (Activation)  (None, 32, 32, 128)          0
['batch_normalization_43[0][0]
                                                                  ']

 batch_normalization_44 (Ba  (None, 32, 32, 128)          512
```

```
['conv2d_58[0][0]']
 tchNormalization)



 conv2d_57 (Conv2D)          (None, 32, 32, 128)          147584
['activation_28[0][0]']


 add_6 (Add)                 (None, 32, 32, 128)          0
['batch_normalization_44[0][0]
                                                                  ',


'conv2d_57[0][0]']


 up_sampling2d_10 (UpSampli  (None, 64, 64, 128)          0
['add_6[0][0]']
 ng2D)



 concatenate_4 (Concatenate  (None, 64, 64, 192)          0
['up_sampling2d_10[0][0]',
 )
'add_1[0][0]']


 batch_normalization_45 (Ba  (None, 64, 64, 192)          768
['concatenate_4[0][0]']
 tchNormalization)



 activation_29 (Activation)  (None, 64, 64, 192)          0
['batch_normalization_45[0][0]
                                                                  ']



 conv2d_59 (Conv2D)          (None, 64, 64, 64)           110656
['activation_29[0][0]']


 batch_normalization_46 (Ba  (None, 64, 64, 64)           256
['conv2d_59[0][0]']
 tchNormalization)
```

```
 conv2d_61 (Conv2D)          (None, 64, 64, 64)           12352
['concatenate_4[0][0]']


 activation_30 (Activation)  (None, 64, 64, 64)           0
['batch_normalization_46[0][0]
                                                                 ']



 batch_normalization_47 (Ba  (None, 64, 64, 64)           256
['conv2d_61[0][0]']
 tchNormalization)



 conv2d_60 (Conv2D)          (None, 64, 64, 64)           36928
['activation_30[0][0]']


 add_7 (Add)                 (None, 64, 64, 64)           0
['batch_normalization_47[0][0]
                                                                 ',


'conv2d_60[0][0]']


 up_sampling2d_11 (UpSampli  (None, 128, 128, 64)         0
['add_7[0][0]']
 ng2D)



 concatenate_5 (Concatenate  (None, 128, 128, 96)         0
['up_sampling2d_11[0][0]',
 )
'add[0][0]']


 batch_normalization_48 (Ba  (None, 128, 128, 96)         384
['concatenate_5[0][0]']
 tchNormalization)



 activation_31 (Activation)  (None, 128, 128, 96)         0
['batch_normalization_48[0][0]
                                                                 ']
```

```
 conv2d_62 (Conv2D)          (None, 128, 128, 32)         27680
['activation_31[0][0]']


 batch_normalization_49 (Ba  (None, 128, 128, 32)         128
['conv2d_62[0][0]']
 tchNormalization)



 conv2d_64 (Conv2D)          (None, 128, 128, 32)         3104
['concatenate_5[0][0]']


 activation_32 (Activation)  (None, 128, 128, 32)         0
['batch_normalization_49[0][0]
                                                                  ']



 batch_normalization_50 (Ba  (None, 128, 128, 32)         128
['conv2d_64[0][0]']
 tchNormalization)



 conv2d_63 (Conv2D)          (None, 128, 128, 32)         9248
['activation_32[0][0]']


 add_8 (Add)                 (None, 128, 128, 32)         0
['batch_normalization_50[0][0]
                                                                  ',


'conv2d_63[0][0]']


 conv2d_65 (Conv2D)          (None, 128, 128, 1)          33
['add_8[0][0]']


==================================================================
============================
Total params: 13026593 (49.69 MB)
Trainable params: 13014817 (49.65 MB)
Non-trainable params: 11776 (46.00 KB)
__________________________________________________________________
_____________________________
```

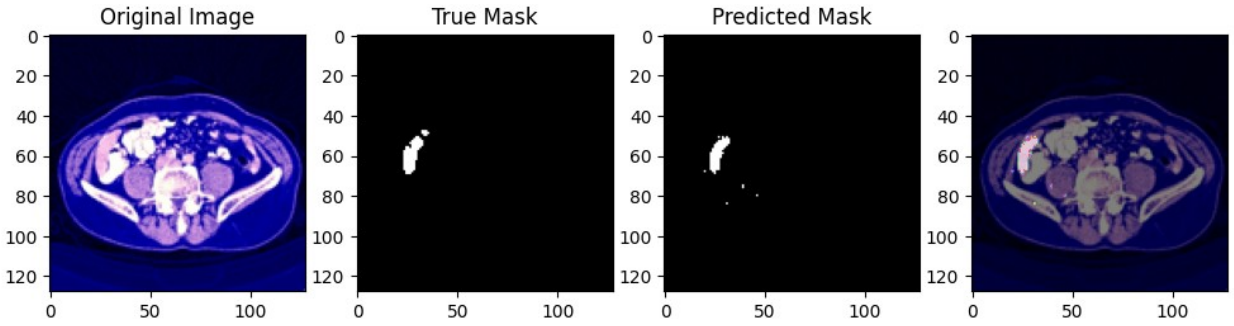
```
model3.compile(optimizer=Adam(lr=0.001, beta_1=0.9,
beta_2=0.999),loss= 'binary_crossentropy', metrics=[dice_coefficient])
model3.save('Resnet_model3.h5')

checkpoint = tf.keras.callbacks.ModelCheckpoint("Resnet_model3.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                   patience=5,
                                                   mode='auto',

restore_best_weights=True)

reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                 factor=0.2,
                                                 patience=2,
                                                 min_lr=1e-6,
                                                 mode='auto',
                                                 verbose=1)

# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]

history3= model3.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model3.save('Resnet_model3.h5')
```

```
Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.0204 -
dice_coefficient: 0.9921
Epoch 1: val_loss improved from inf to 0.23601, saving model to
Resnet_model3.h5
394/394 [==============================] - 60s 96ms/step - loss:
0.0204 - dice_coefficient: 0.9921 - val_loss: 0.2360 -
val_dice_coefficient: 0.9454 - lr: 0.0010
Epoch 2/10
394/394 [==============================] - ETA: 0s - loss: 0.0109 -
dice_coefficient: 0.9944
Epoch 2: val_loss did not improve from 0.23601
394/394 [==============================] - 33s 84ms/step - loss:
0.0109 - dice_coefficient: 0.9944 - val_loss: 9.7734 -
val_dice_coefficient: 0.8504 - lr: 0.0010
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: -0.0050 -
dice_coefficient: 0.9936
Epoch 3: val_loss improved from 0.23601 to 0.02568, saving model to
Resnet_model3.h5
```

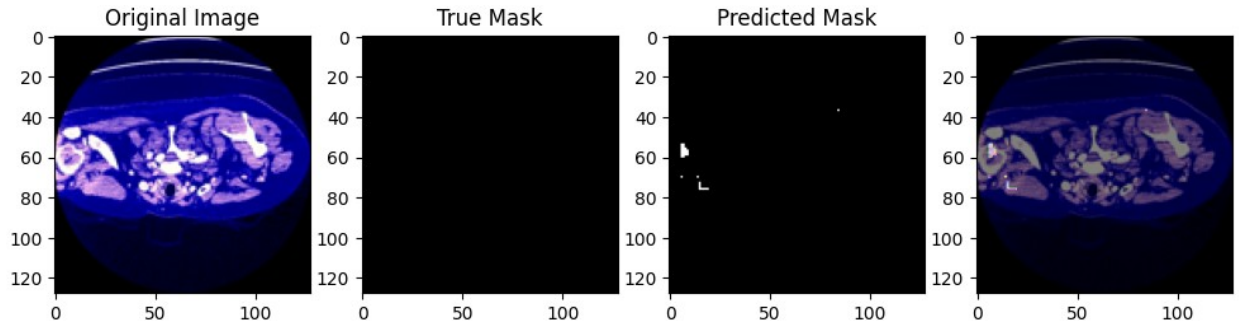
```
394/394 [==============================] - 34s 86ms/step - loss: -
0.0050 - dice_coefficient: 0.9936 - val_loss: 0.0257 -
val_dice_coefficient: 0.9953 - lr: 0.0010
Epoch 4/10
394/394 [==============================] - ETA: 0s - loss: 0.0090 -
dice_coefficient: 0.9942
Epoch 4: val_loss improved from 0.02568 to 0.01697, saving model to
Resnet_model3.h5
394/394 [==============================] - 34s 86ms/step - loss:
0.0090 - dice_coefficient: 0.9942 - val_loss: 0.0170 -
val_dice_coefficient: 0.9905 - lr: 0.0010
Epoch 5/10
394/394 [==============================] - ETA: 0s - loss: 0.0020 -
dice_coefficient: 0.9942
Epoch 5: val_loss did not improve from 0.01697
394/394 [==============================] - 33s 84ms/step - loss:
0.0020 - dice_coefficient: 0.9942 - val_loss: 0.0542 -
val_dice_coefficient: 0.9949 - lr: 0.0010
Epoch 6/10
394/394 [==============================] - ETA: 0s - loss: -0.0190 -
dice_coefficient: 0.9926
Epoch 6: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.

Epoch 6: val_loss did not improve from 0.01697
394/394 [==============================] - 33s 84ms/step - loss: -
0.0190 - dice_coefficient: 0.9926 - val_loss: 0.0205 -
val_dice_coefficient: 0.9961 - lr: 0.0010
Epoch 7/10
394/394 [==============================] - ETA: 0s - loss: 0.0125 -
dice_coefficient: 0.9953
Epoch 7: val_loss improved from 0.01697 to 0.01024, saving model to
Resnet_model3.h5
394/394 [==============================] - 34s 86ms/step - loss:
0.0125 - dice_coefficient: 0.9953 - val_loss: 0.0102 -
val_dice_coefficient: 0.9960 - lr: 2.0000e-04
Epoch 8/10
394/394 [==============================] - ETA: 0s - loss: 0.0097 -
dice_coefficient: 0.9959
Epoch 8: val_loss improved from 0.01024 to 0.00931, saving model to
Resnet_model3.h5
394/394 [==============================] - 34s 86ms/step - loss:
0.0097 - dice_coefficient: 0.9959 - val_loss: 0.0093 -
val_dice_coefficient: 0.9953 - lr: 2.0000e-04
Epoch 9/10
394/394 [==============================] - ETA: 0s - loss: 0.0022 -
dice_coefficient: 0.9962
Epoch 9: val_loss improved from 0.00931 to -0.01670, saving model to
Resnet_model3.h5
```

```
394/394 [==============================] - 34s 86ms/step - loss:
0.0022 - dice_coefficient: 0.9962 - val_loss: -0.0167 -
val_dice_coefficient: 0.9906 - lr: 2.0000e-04
Epoch 10/10
394/394 [==============================] - ETA: 0s - loss: -0.6425 -
dice_coefficient: 0.9951
Epoch 10: val_loss improved from -0.01670 to -0.53361, saving model to
Resnet_model3.h5
394/394 [==============================] - 34s 86ms/step - loss: -
0.6425 - dice_coefficient: 0.9951 - val_loss: -0.5336 -
val_dice_coefficient: 0.9917 - lr: 2.0000e-04

scores3 = model3.evaluate(x_valid, y_valid)
scores3[1]

57/57 [==============================] - 4s 43ms/step - loss: -0.5336
- dice_coefficient: 0.9917

0.9916637539863586

prediction3 = model3.predict(x_test)

29/29 [==============================] - 2s 59ms/step

test_scores3 = model3.evaluate(x_test, y_test)
test_scores3[1]

29/29 [==============================] - 1s 42ms/step - loss: -0.3832
- dice_coefficient: 0.9915

0.9914811253547668

import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 50, replace=False)
for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model3
    predicted_mask = model3.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
```

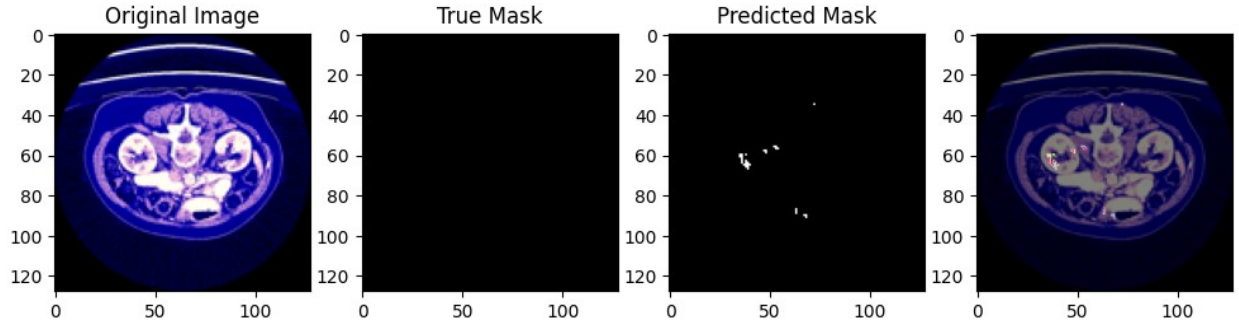
```
    threshold = 0.5  # Adjust this threshold based on your model's
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')

    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
'nipy_spectral')

    plt.show()

333
1/1 [==============================] - 0s 406ms/step
```



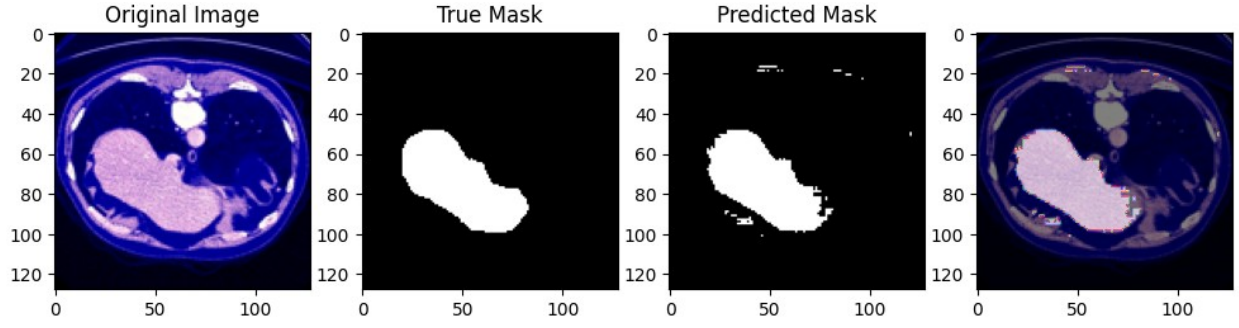

```
812
1/1 [==============================] - 0s 23ms/step
```



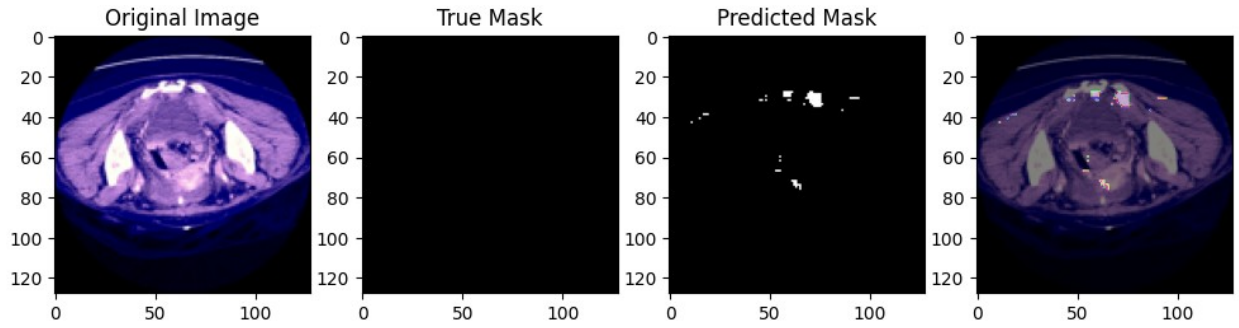

```
745
1/1 [==============================] - 0s 23ms/step
```



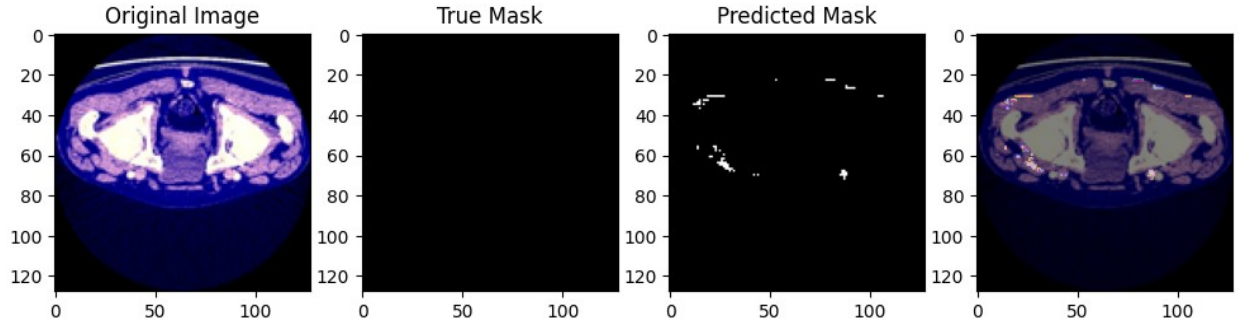

```
245
1/1 [==============================] - 0s 25ms/step
```



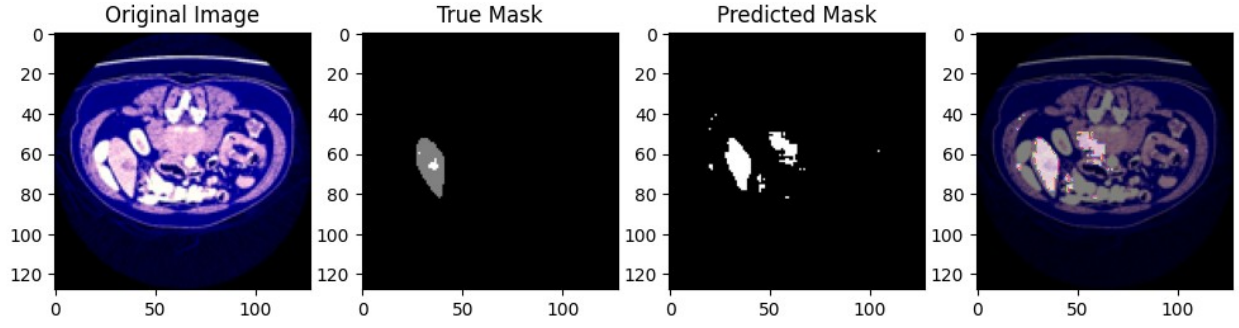

```
40
1/1 [==============================] - 0s 23ms/step
```



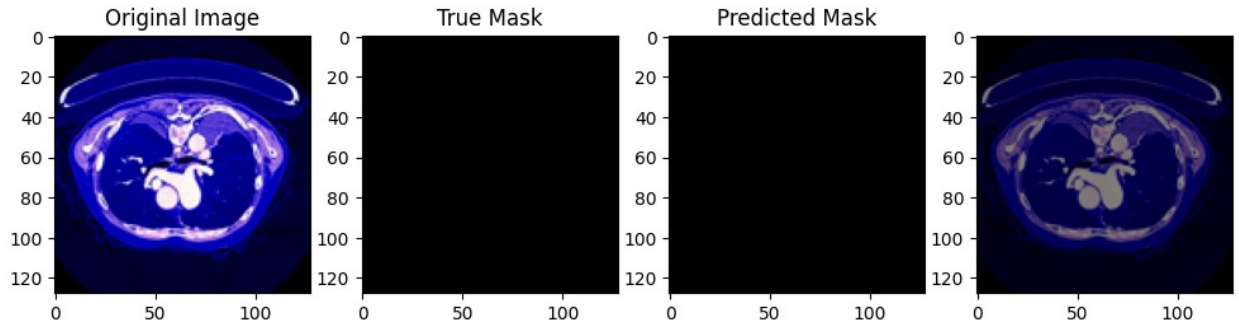

```
427
1/1 [==============================] - 0s 23ms/step
```



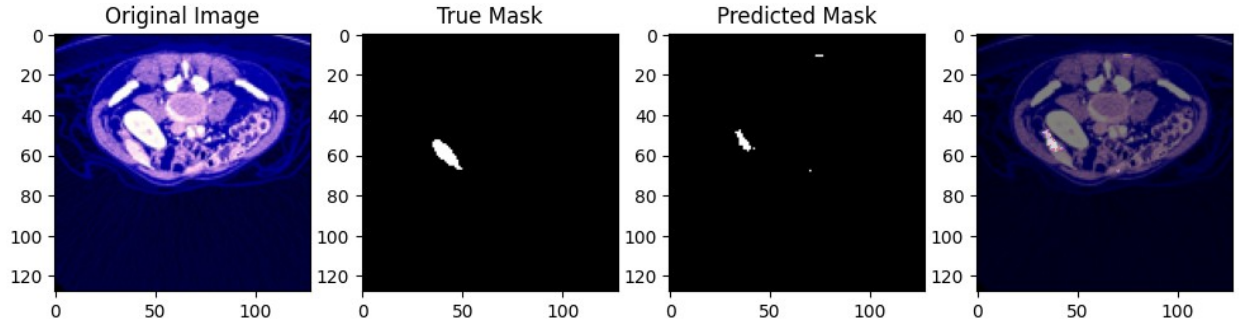

```
303
1/1 [==============================] - 0s 24ms/step
```



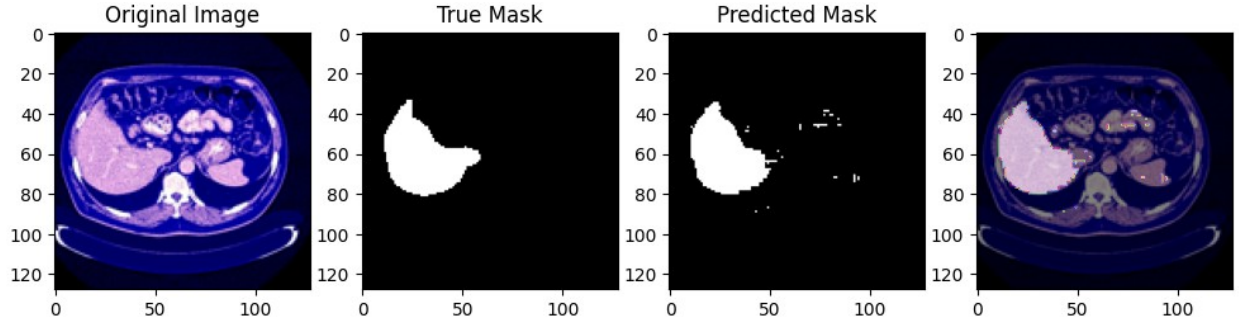

```
588
1/1 [==============================] - 0s 23ms/step
```



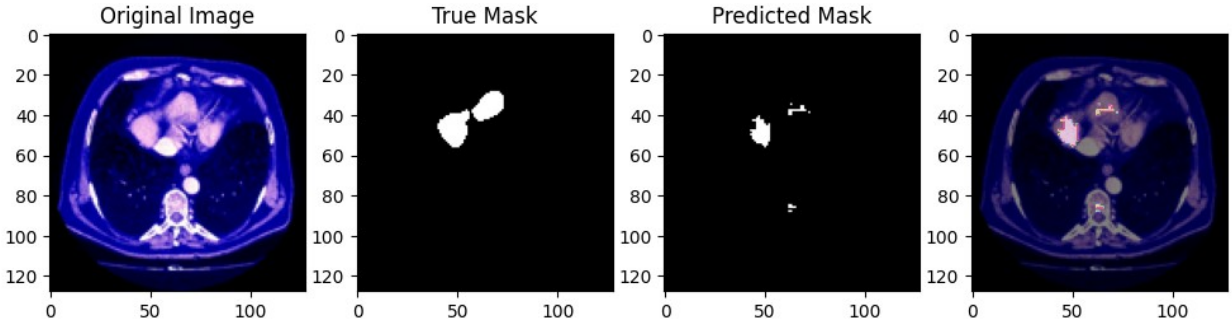

```
210
1/1 [==============================] - 0s 23ms/step
```



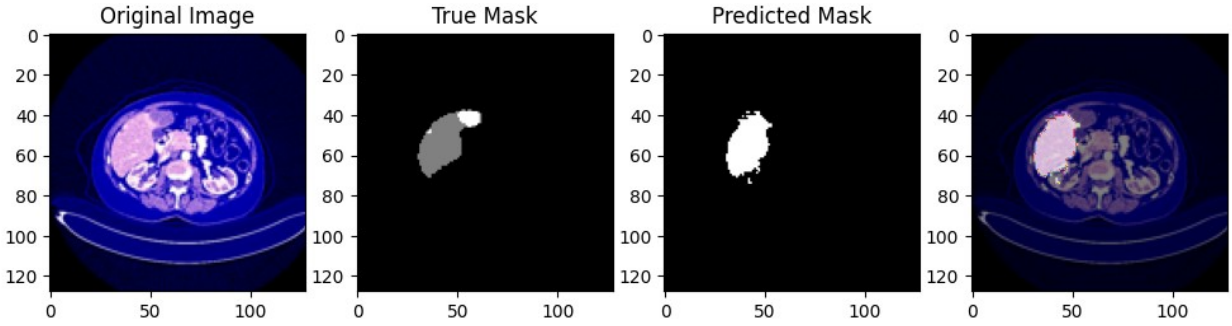

```
137
1/1 [==============================] - 0s 22ms/step
```



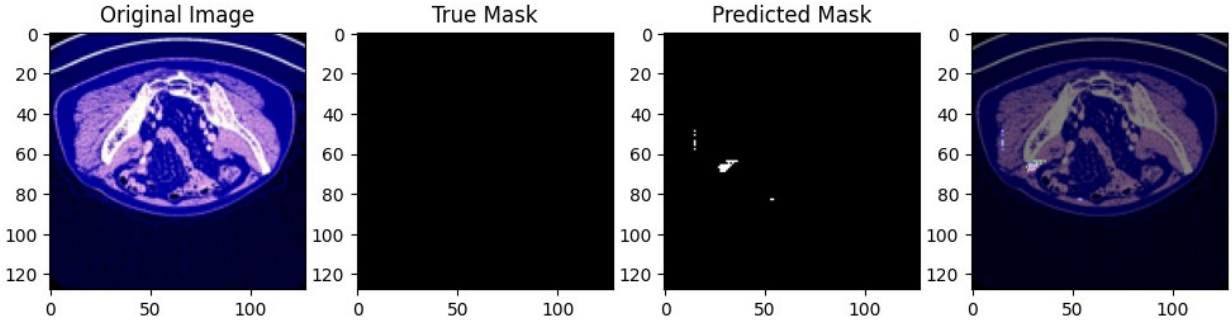

```
138
1/1 [==============================] - 0s 23ms/step
```



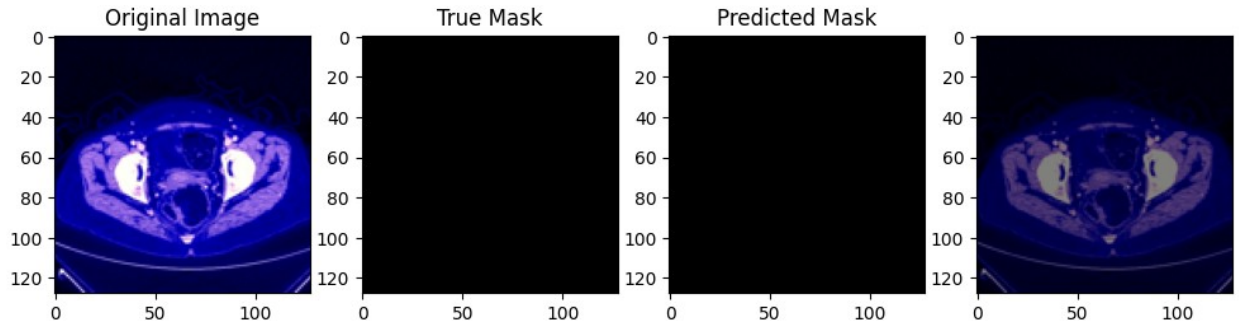

```
261
1/1 [==============================] - 0s 24ms/step
```



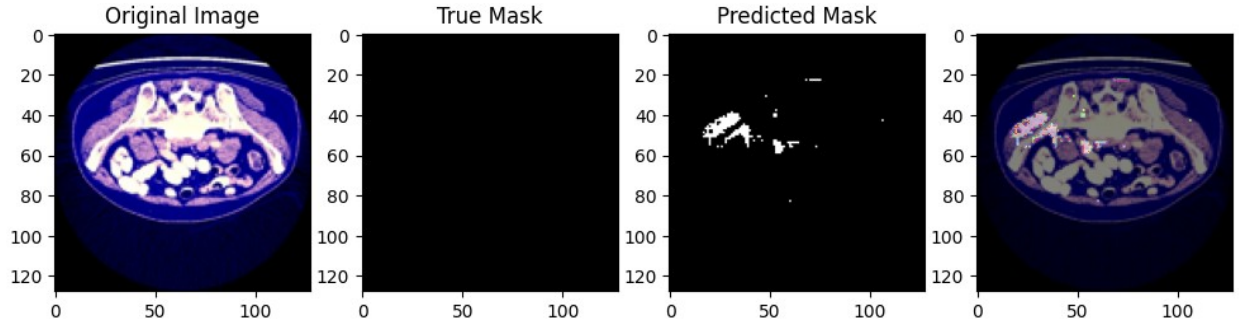

```
352
1/1 [==============================] - 0s 22ms/step
```



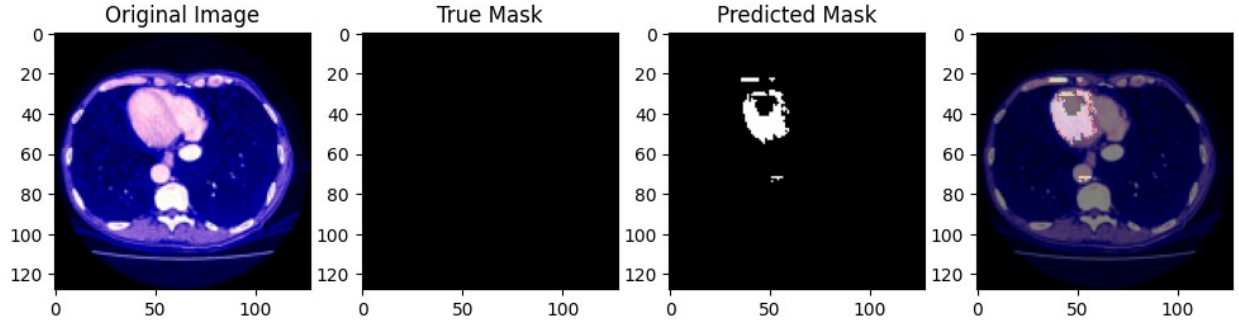

```
360
1/1 [==============================] - 0s 25ms/step
```



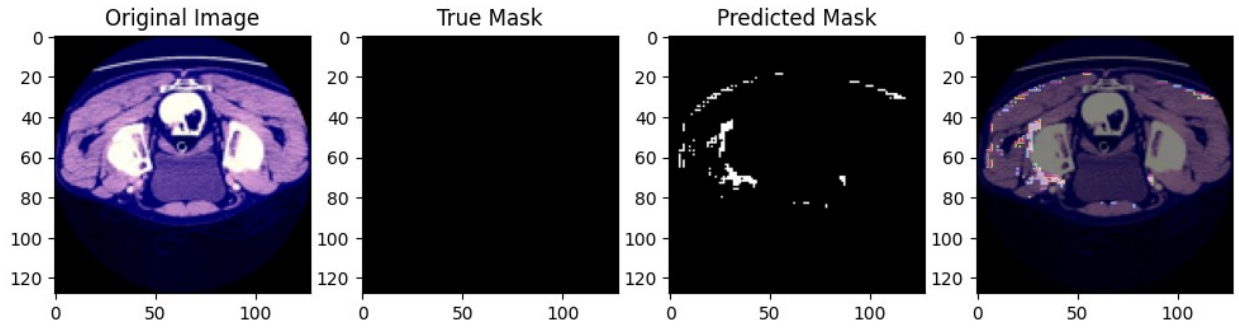

```
347
1/1 [==============================] - 0s 23ms/step
```



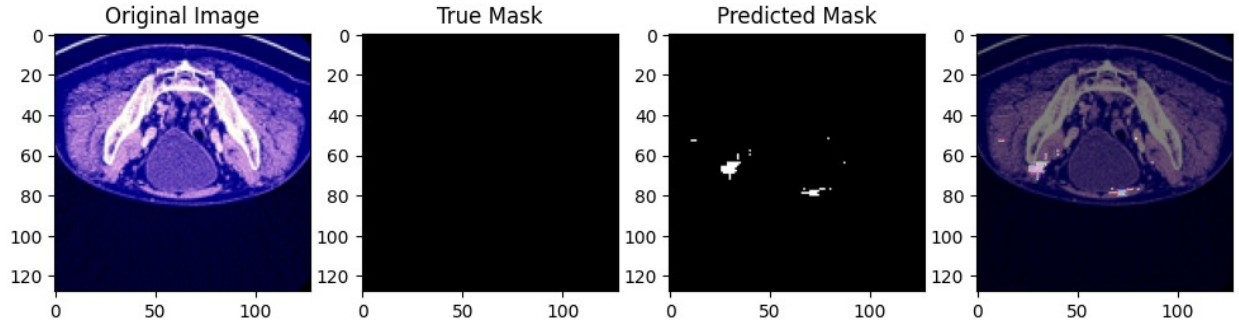

```
479
1/1 [==============================] - 0s 23ms/step
```



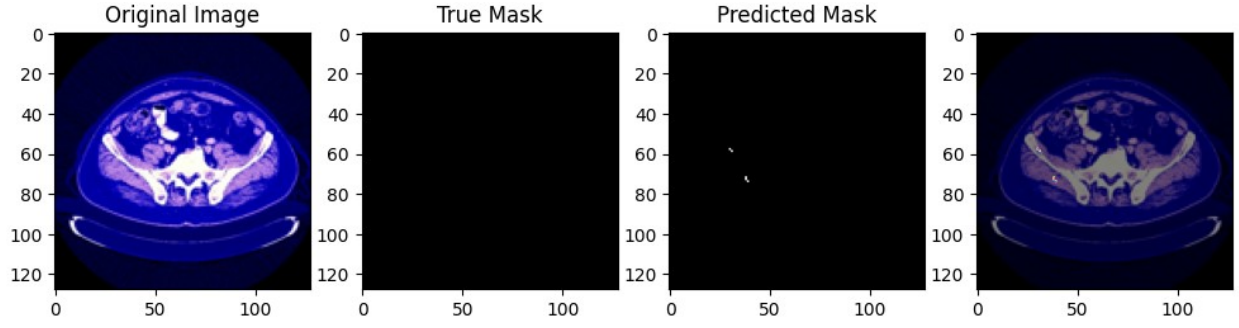

```
307
1/1 [==============================] - 0s 25ms/step
```



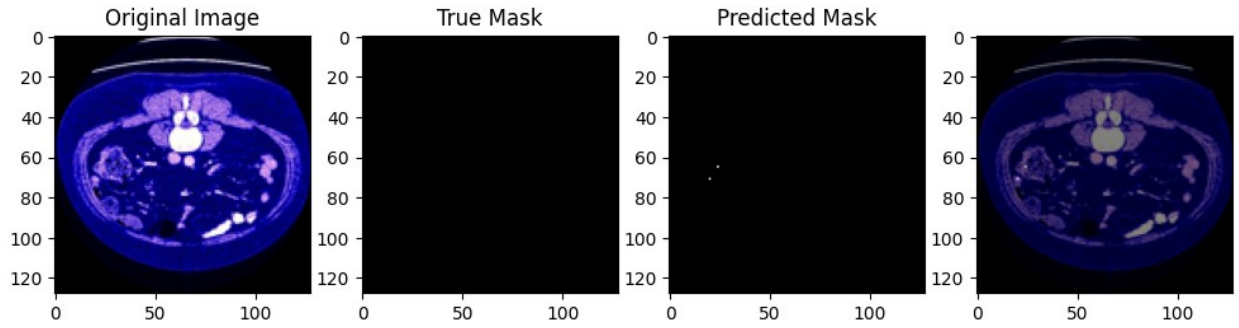

```
658
1/1 [==============================] - 0s 25ms/step
```



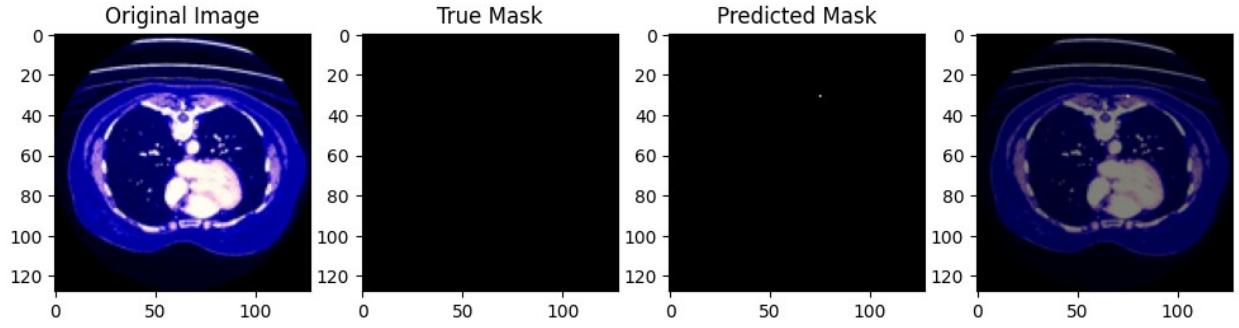

```
895
1/1 [==============================] - 0s 24ms/step
```



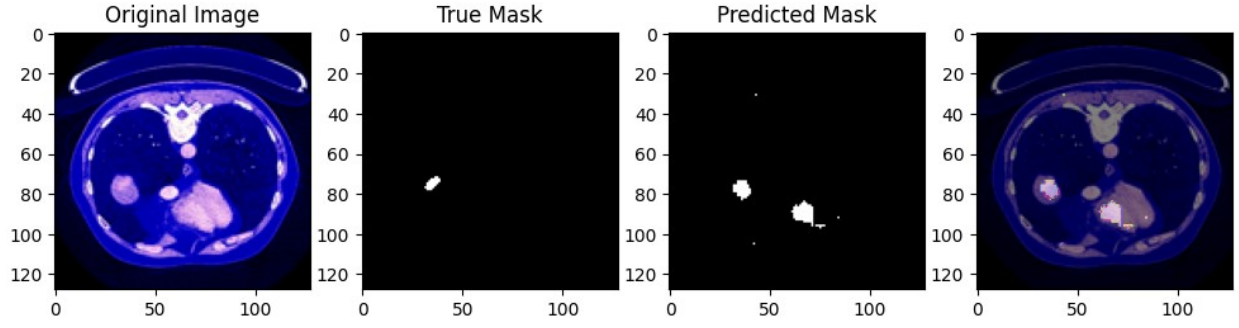

```
111
1/1 [==============================] - 0s 23ms/step
```



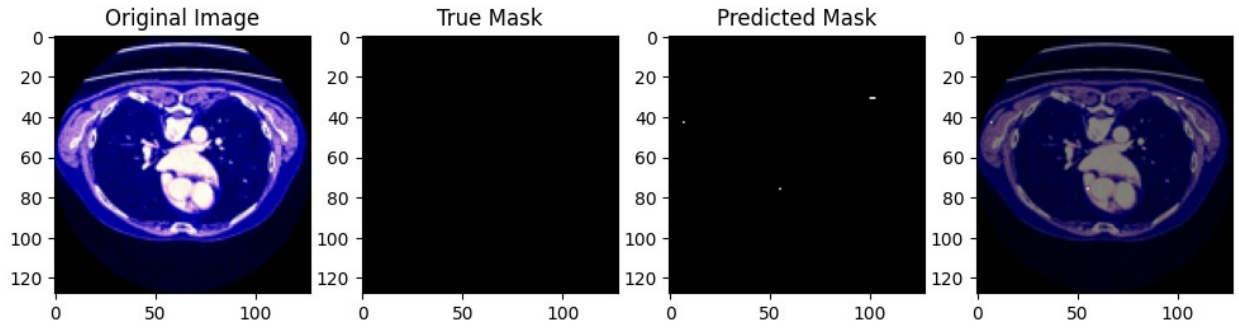

```
297
1/1 [==============================] - 0s 24ms/step
```



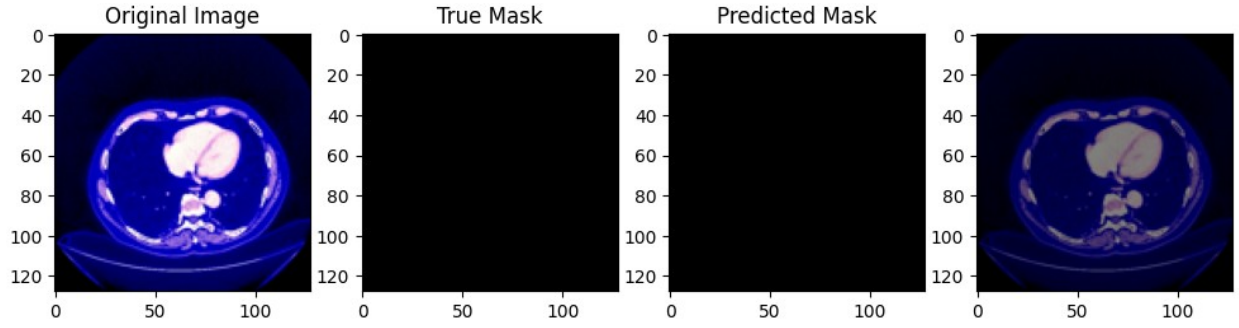

```
452
1/1 [==============================] - 0s 24ms/step
```



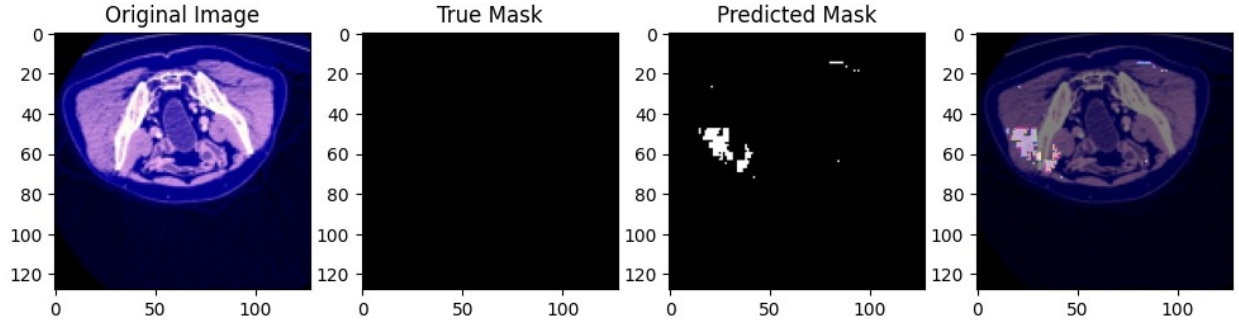

```
193
1/1 [==============================] - 0s 23ms/step
```



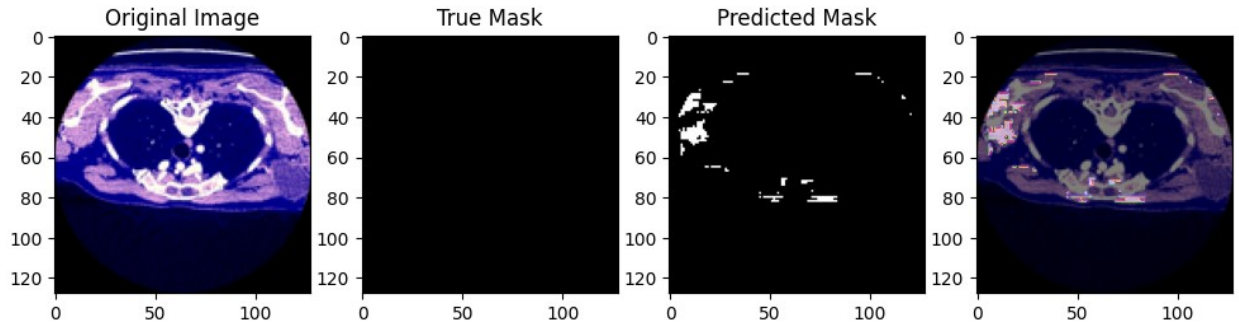

```
863
1/1 [==============================] - 0s 23ms/step
```



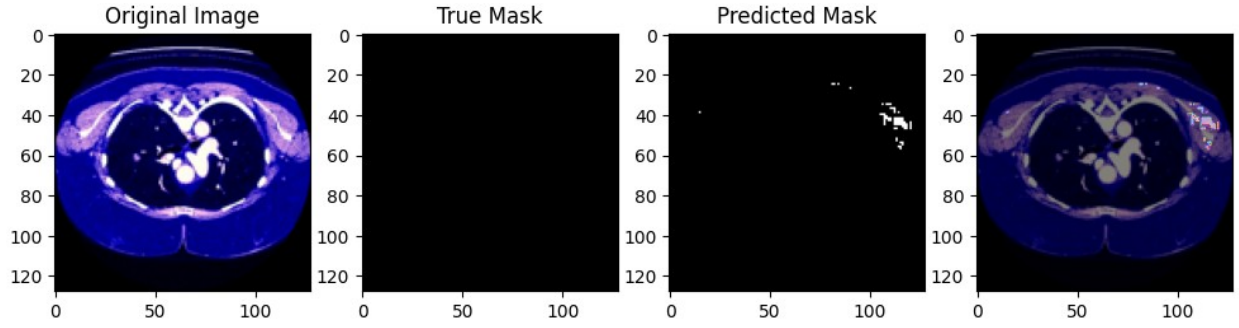

```
542
1/1 [==============================] - 0s 24ms/step
```



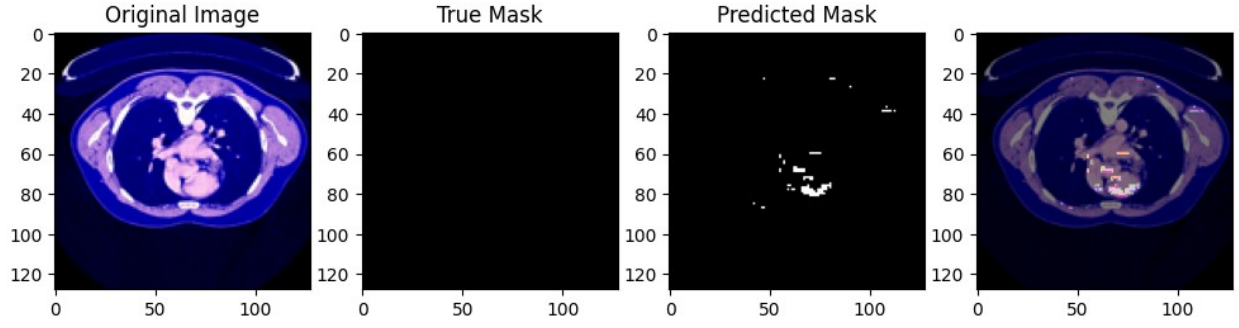

```
855
1/1 [==============================] - 0s 23ms/step
```



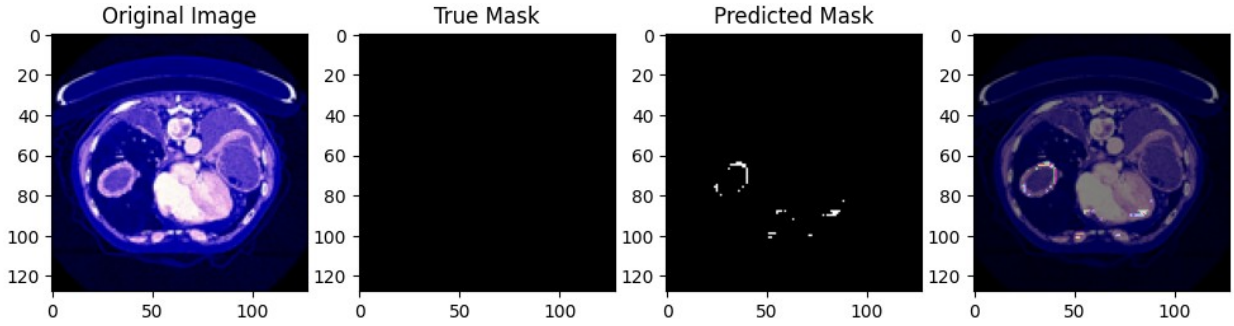

```
31
1/1 [==============================] - 0s 23ms/step
```



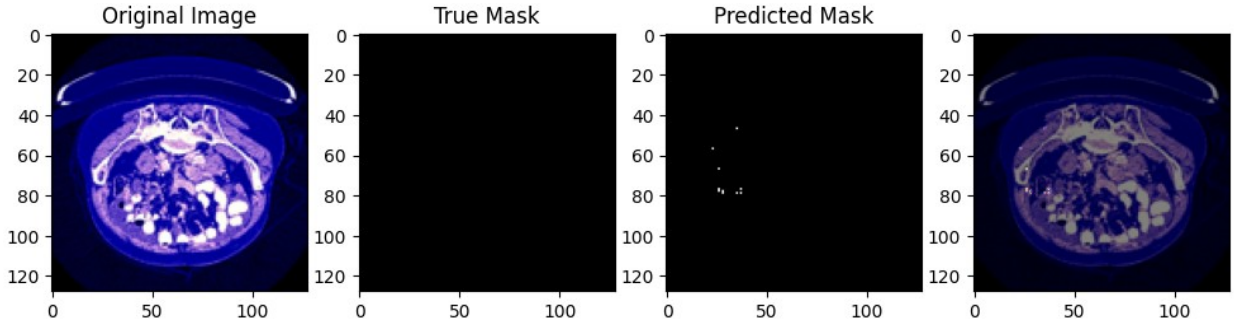

```
818
1/1 [==============================] - 0s 23ms/step
```



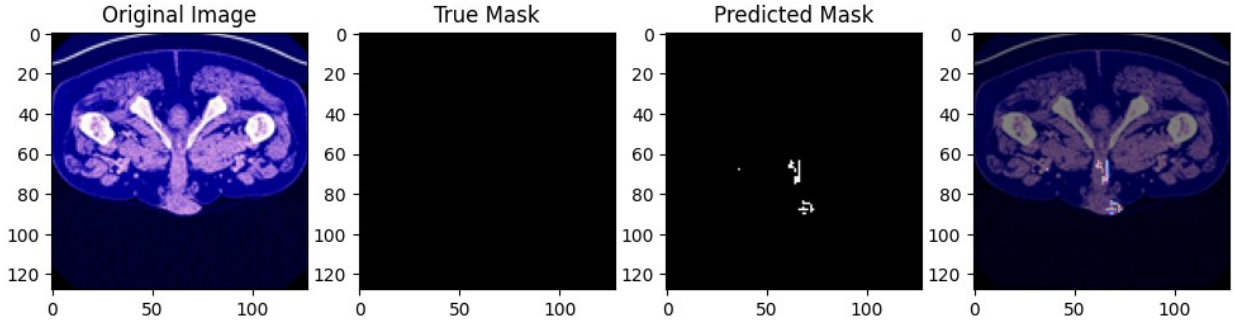

```
64
1/1 [==============================] - 0s 23ms/step
```



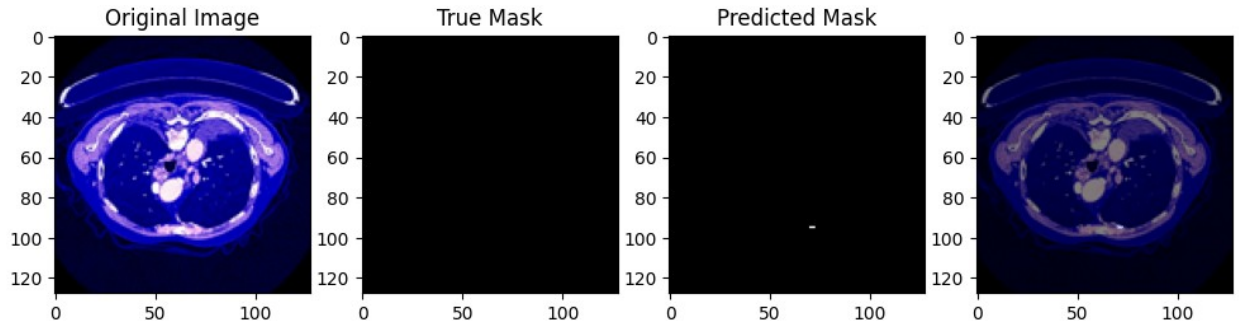

```
709
1/1 [==============================] - 0s 23ms/step
```



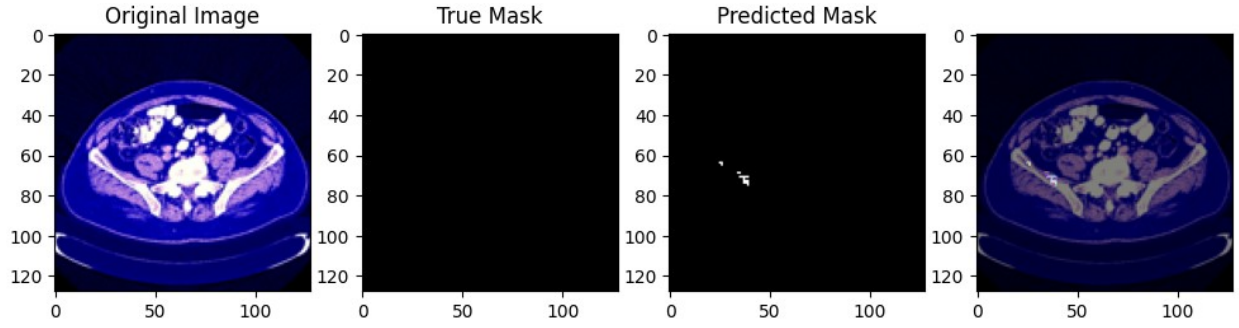

```
260
1/1 [==============================] - 0s 32ms/step
```



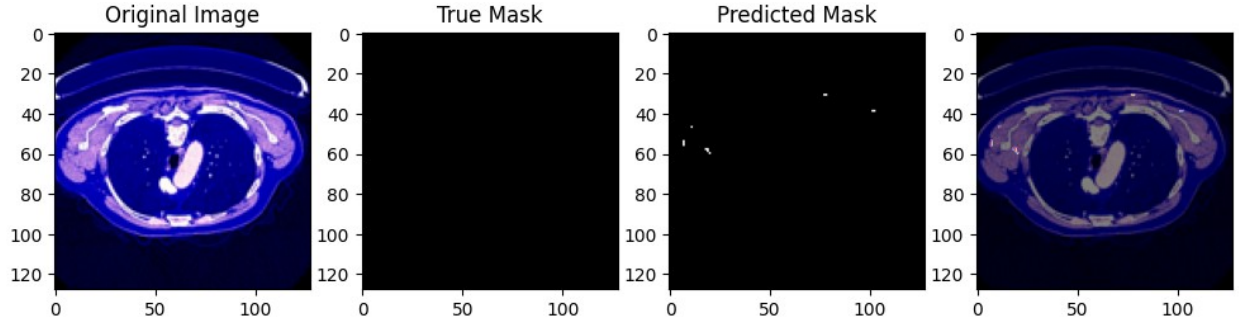

```
142
1/1 [==============================] - 0s 23ms/step
```



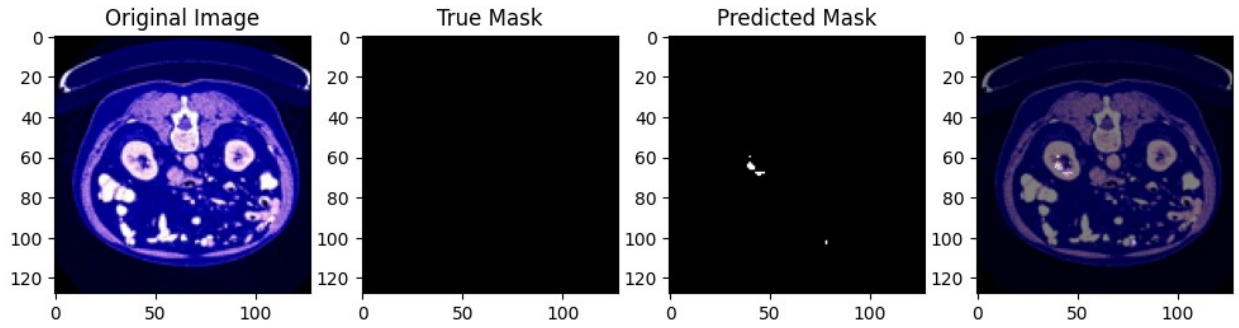

```
631
1/1 [==============================] - 0s 24ms/step
```



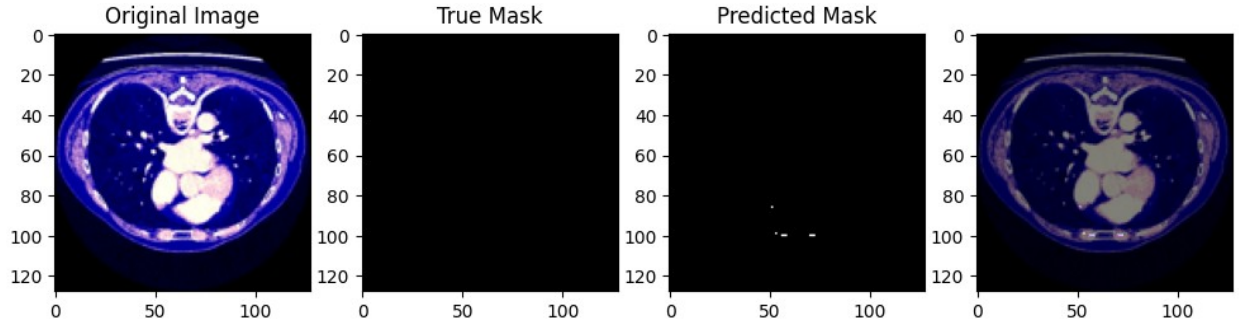

```
430
1/1 [==============================] - 0s 24ms/step
```



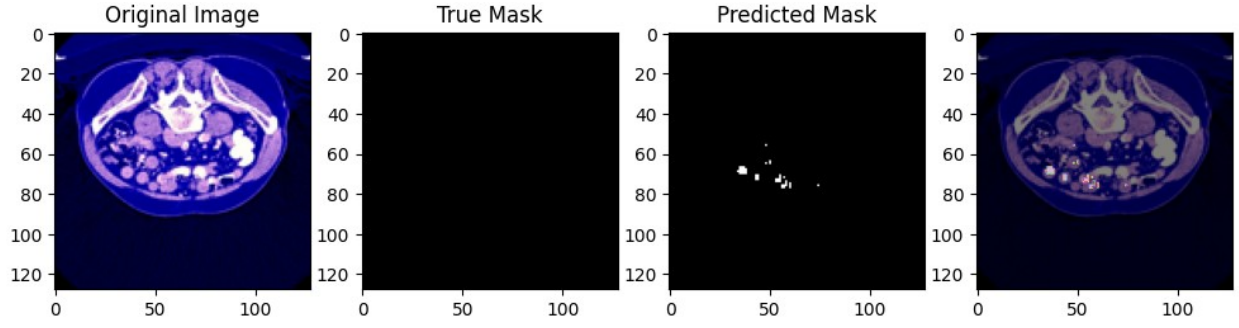

```
328
1/1 [==============================] - 0s 23ms/step
```



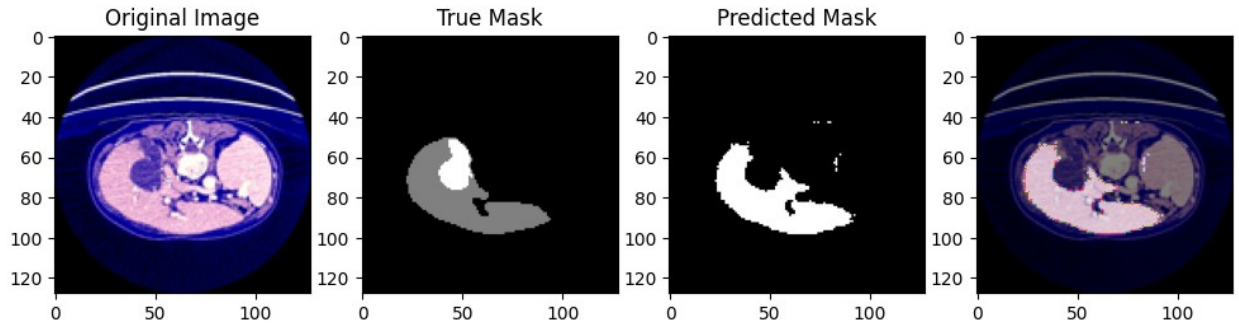

```
626
1/1 [==============================] - 0s 24ms/step
```



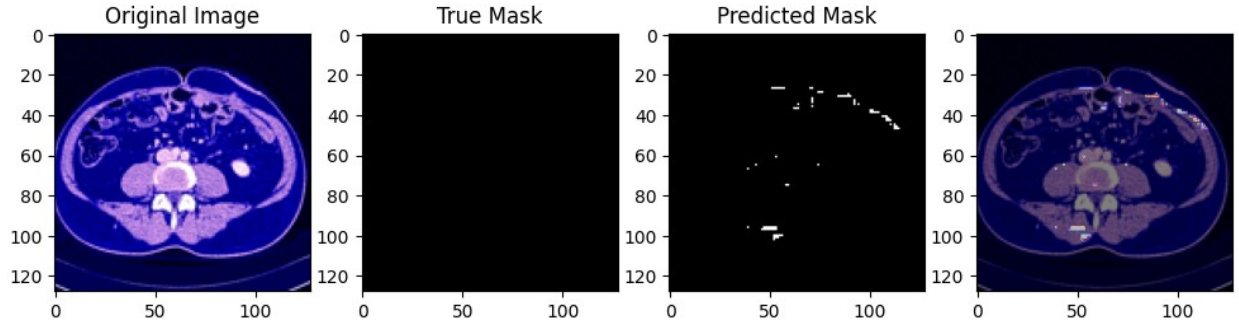

```
24
1/1 [==============================] - 0s 24ms/step
```



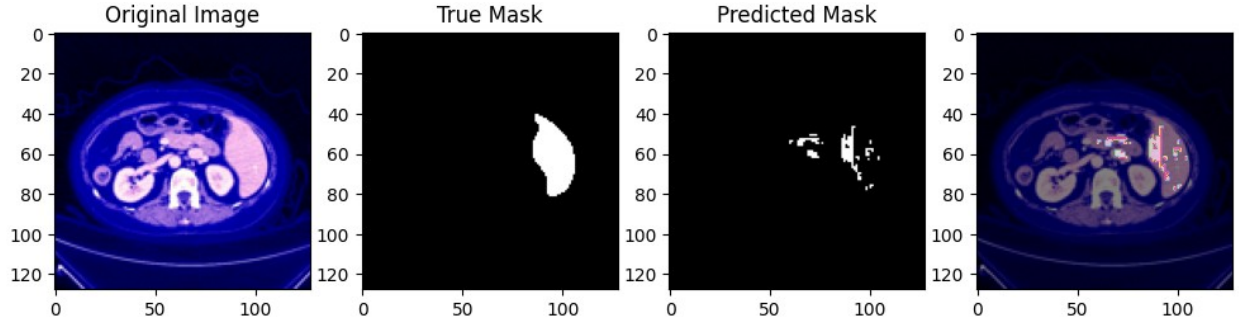

```
199
1/1 [==============================] - 0s 24ms/step
```



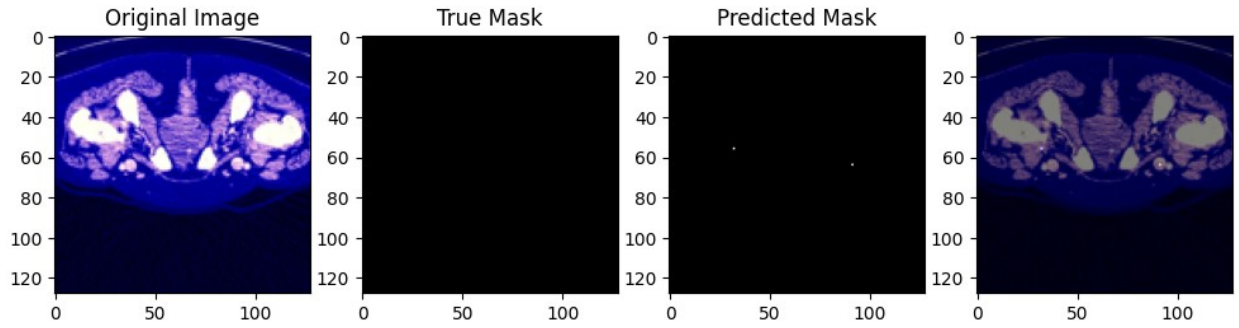

```
879
1/1 [==============================] - 0s 23ms/step
```



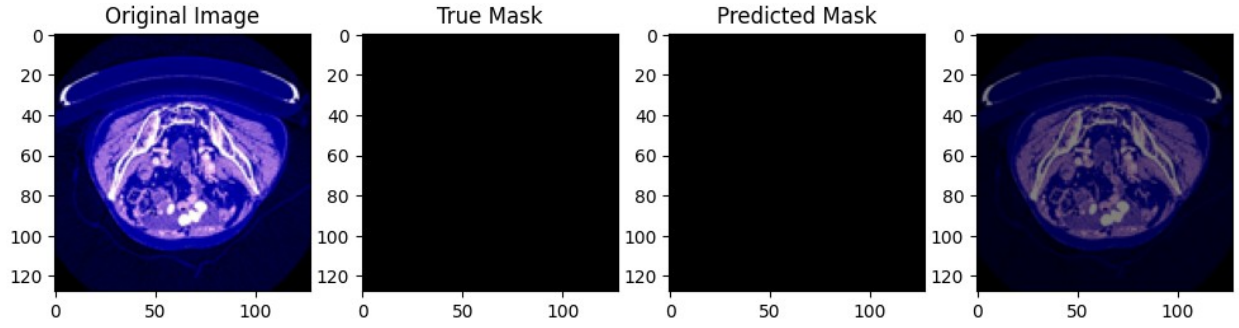

```
891
1/1 [==============================] - 0s 25ms/step
```



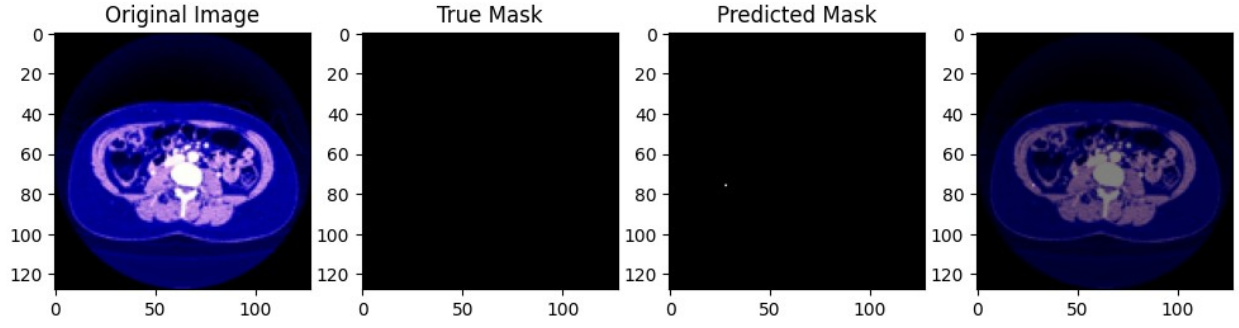

```
121
1/1 [==============================] - 0s 23ms/step
```



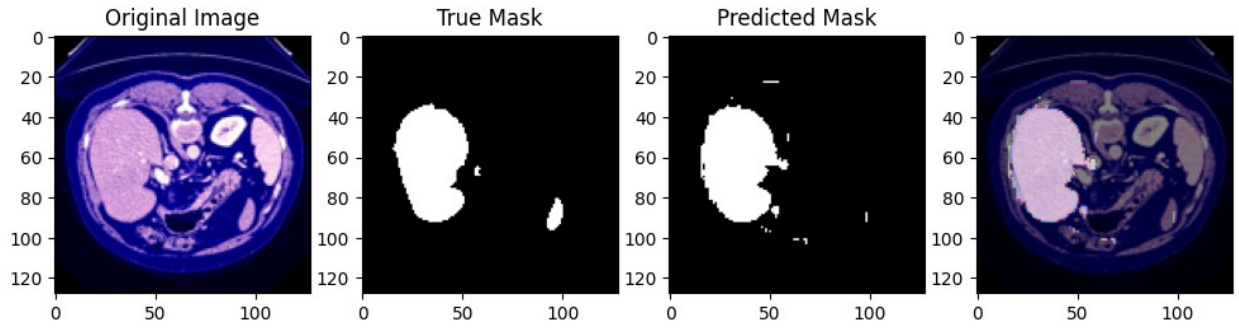

```
339
1/1 [==============================] - 0s 22ms/step
```



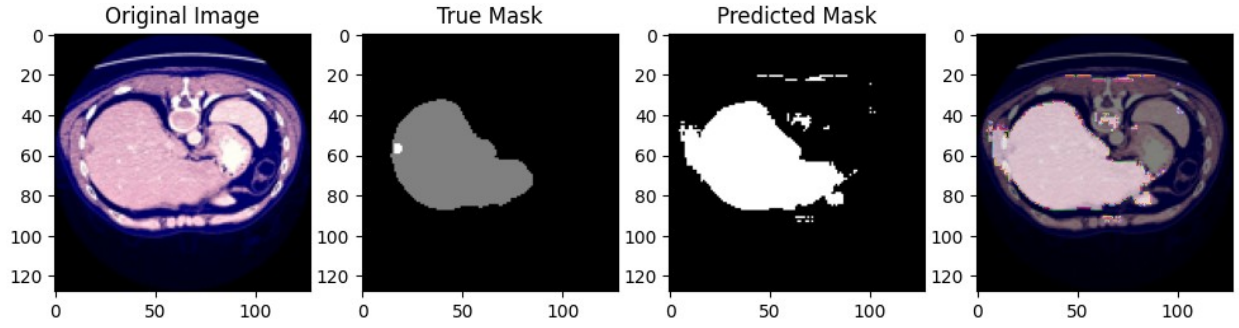

```
652
1/1 [==============================] - 0s 23ms/step
```



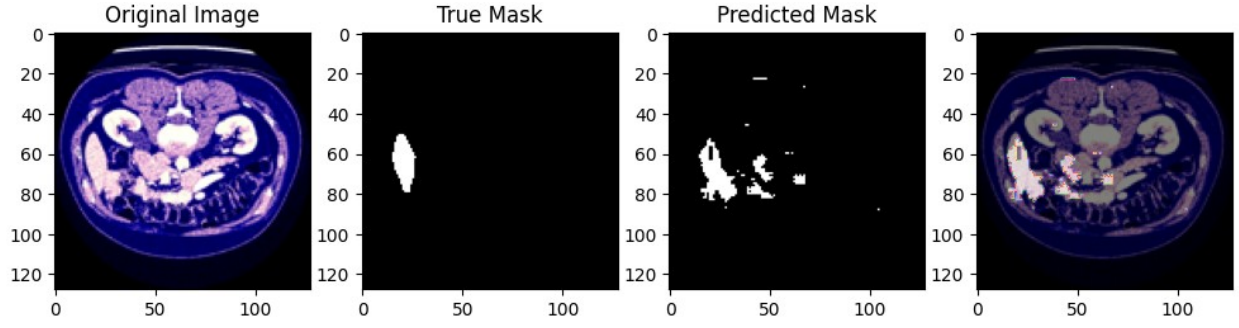

```
744
1/1 [==============================] - 0s 29ms/step
```



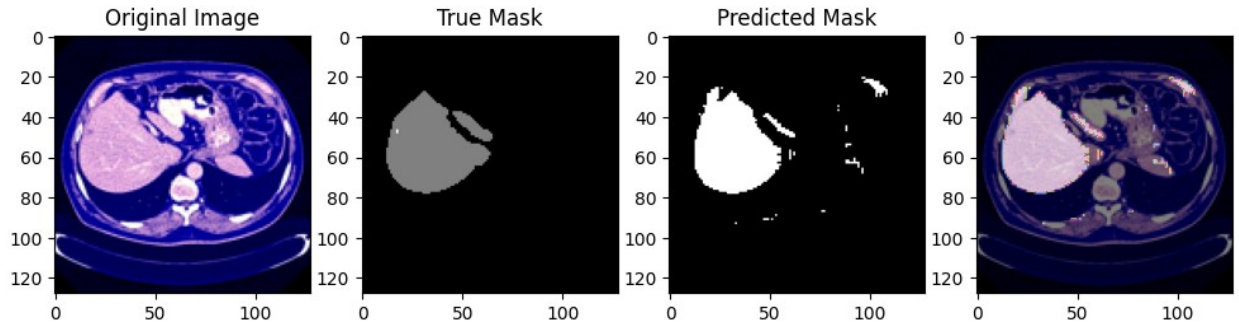

```
366
1/1 [==============================] - 0s 23ms/step
```



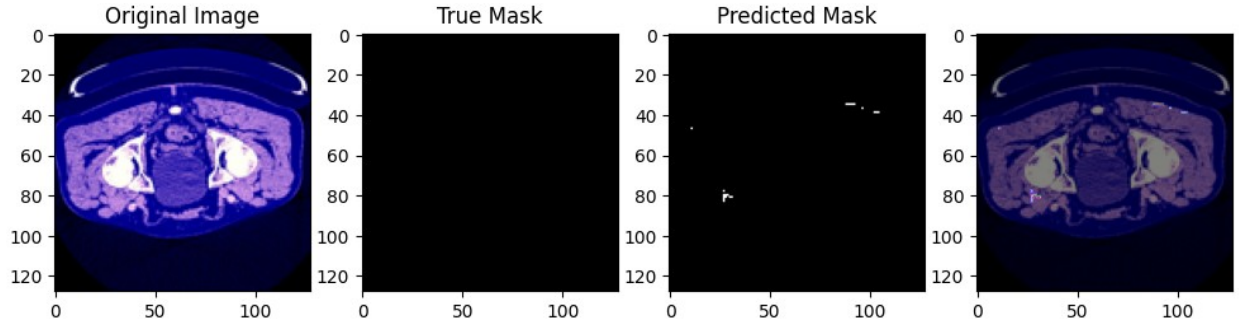

```
87
1/1 [==============================] - 0s 23ms/step
```



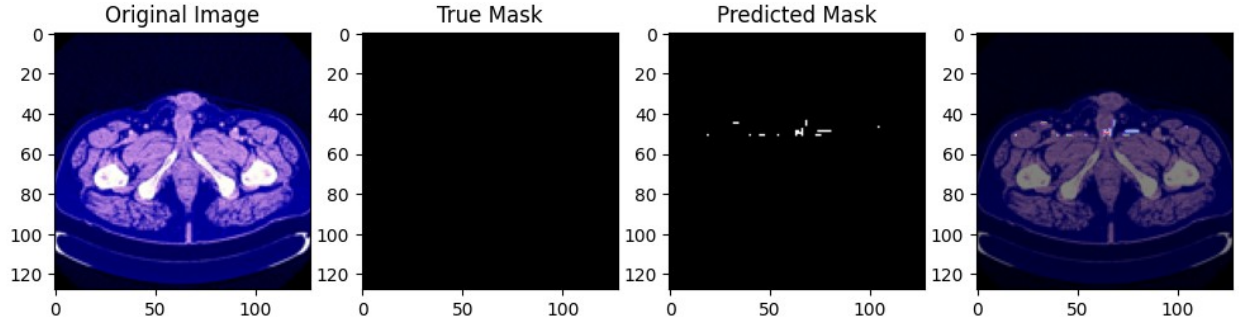

```
68
1/1 [==============================] - 0s 25ms/step
```



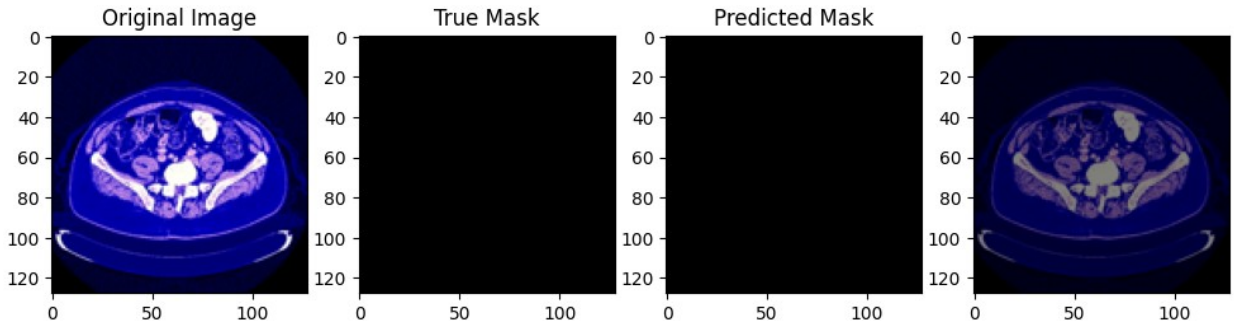

```
353
1/1 [==============================] - 0s 24ms/step
```



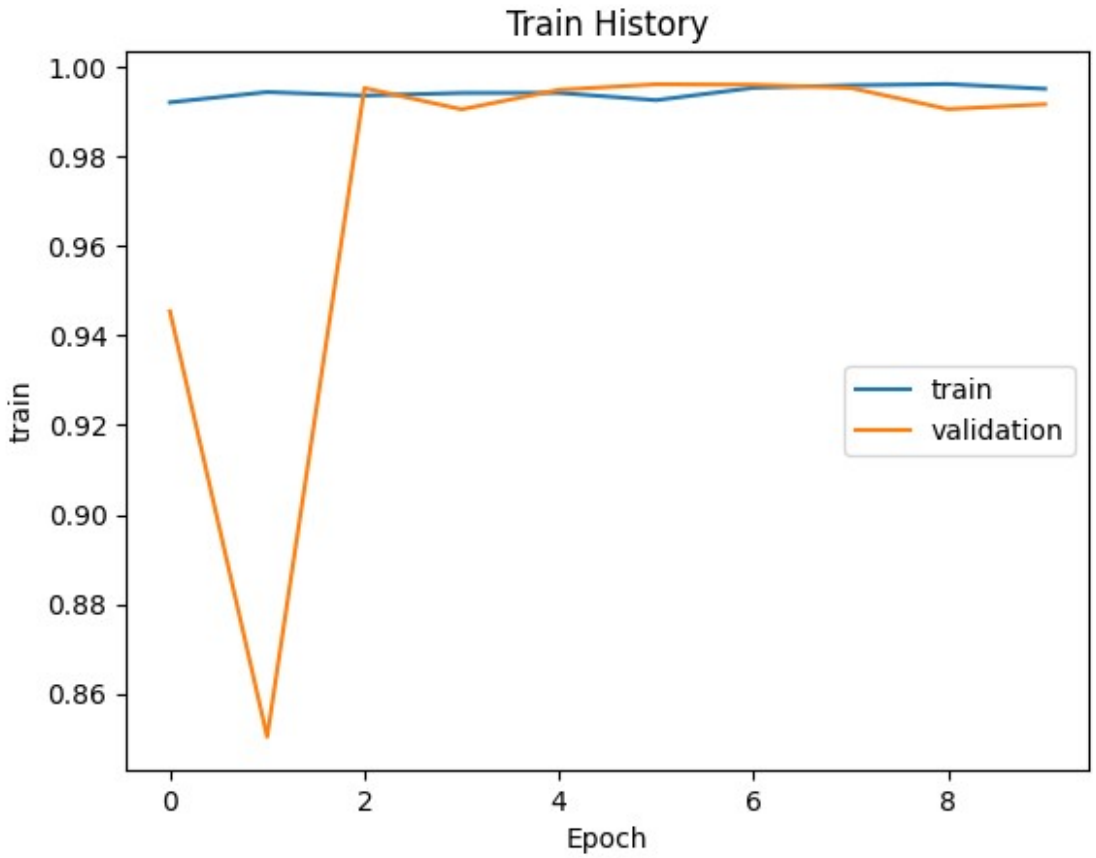

```
212
1/1 [==============================] - 0s 22ms/step
```



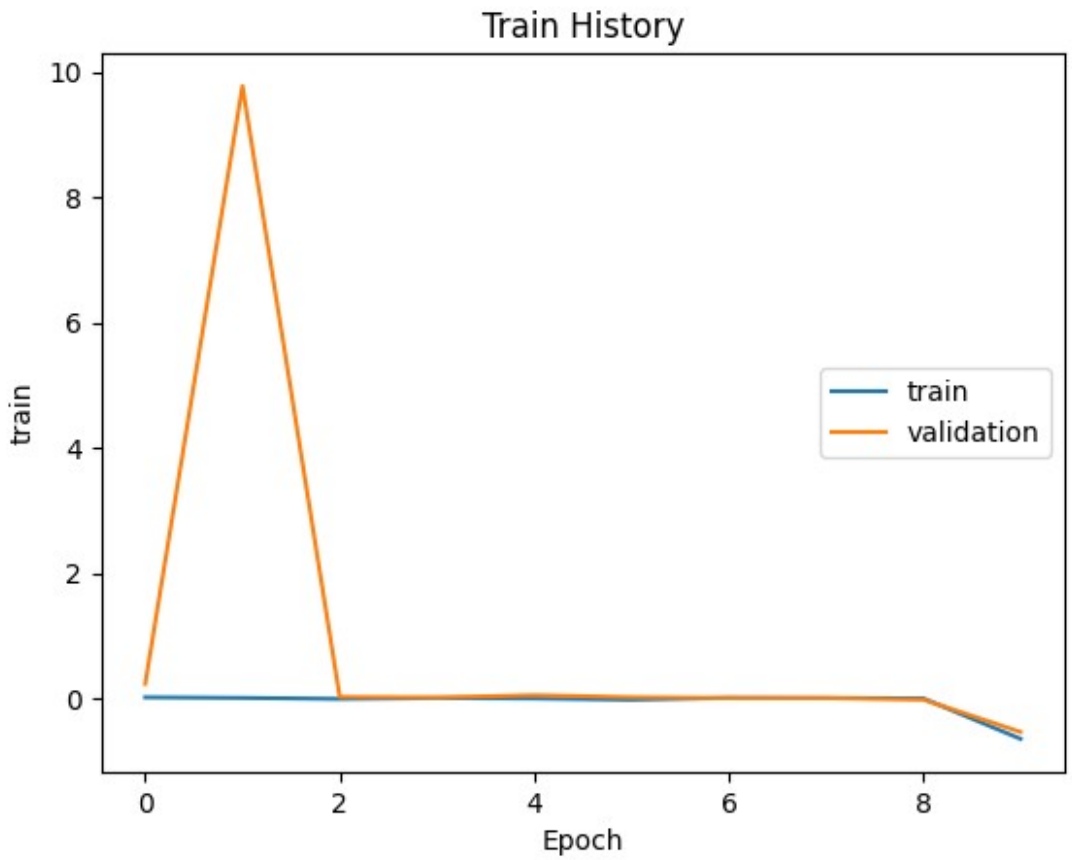

```
741
1/1 [==============================] - 0s 23ms/step
```




```
show_history(history3, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history3, 'loss', 'val_loss')
```




```
import tensorflow as tf
from tensorflow import keras

def conv_block(inputs, filters, kernel_size, strides):
  """
  This function defines a convolutional block with ReLU activation and
BatchNormalization.
  """
  x = keras.layers.Conv2D(filters, kernel_size, strides=strides,
padding='same', activation='relu')(inputs)
  x = keras.layers.BatchNormalization()(x)
  return x

def residual_block(inputs, filters, kernel_size=(3, 3), strides=(1,
1)):
  """
  This function defines a residual block with skip connection.
  """
  x = conv_block(inputs, filters, kernel_size, strides)
  x = keras.layers.Conv2D(filters, kernel_size, padding='same')(x)
  x = keras.layers.BatchNormalization()(x)

  shortcut = keras.layers.Conv2D(filters, (1, 1), strides=strides)
```

```
(inputs)
  shortcut = keras.layers.BatchNormalization()(shortcut)

  x = keras.layers.add([x, shortcut])
  x = keras.layers.Activation('relu')(x)
  return x

def PSPModule(inputs, filters):
  """
  This function defines the Pyramid Pooling Module with global average
pooling and 1x1 convolution.
  """
  branches = []

  # Global Average Pooling
  branch = keras.layers.GlobalAveragePooling2D()(inputs)
  branch = keras.layers.Reshape((1, 1, filters))(branch)
  # Upsample to match the size of the other branch
  branch = keras.layers.UpSampling2D(size=(inputs.shape[1],
inputs.shape[2]), interpolation='bilinear')(branch)
  branch = keras.layers.Conv2D(filters, (1, 1), padding='same',
activation='relu')(branch)
  branches.append(branch)

  # 1x1 convolution branch
  branch = keras.layers.Conv2D(filters, (1, 1), padding='same',
activation='relu')(inputs)
  branches.append(branch)

  return keras.layers.concatenate(branches)


def PSPNet(input_shape=(128, 128, 3), num_classes=1):
  """
  This function defines the PSPNet model with encoder, PSP module,
decoder, and output layer.
  """
  inputs = keras.layers.Input(input_shape)

  # Encoder
  x = conv_block(inputs, 64, (3, 3), strides=(2, 2))
  x = residual_block(x, 64, (3, 3))
  x = residual_block(x, 128, (3, 3), strides=(2, 2))
  x = residual_block(x, 128, (3, 3))
  x = residual_block(x, 256, (3, 3), strides=(2, 2))
  x = residual_block(x, 256, (3, 3))
  x = residual_block(x, 512, (3, 3), strides=(2, 2))

  # PSP Module
  x = PSPModule(x, 512)  # Assuming 512 filters in the last residual
```

```
block

  # Decoder
  x = conv_block(x, 512, (3, 3), strides=(1, 1))
  x = keras.layers.UpSampling2D(size=(2, 2), interpolation='bilinear')
(x)
  x = conv_block(x, 256, (3, 3), strides=(1, 1))
  x = keras.layers.UpSampling2D(size=(2, 2), interpolation='bilinear')
(x)
  x = conv_block(x, 128, (3, 3), strides=(1, 1))
  x = keras.layers.UpSampling2D(size=(2, 2), interpolation='bilinear')
(x)
  x = conv_block(x, 64, (3, 3), strides=(1, 1))

  x = keras.layers.UpSampling2D(size=(2, 2), interpolation='bilinear')
(x)
  # Output - Assuming multi-class segmentation
(categorical_crossentropy)
  outputs = keras.layers.Conv2D(num_classes, (1, 1),
activation='sigmoid')(x)
  model = keras.models.Model(inputs, outputs)
  return model
# Instantiate the model
model4 = PSPNet()
# Compile the model
model4.compile(optimizer='Adam', loss='binary_crossentropy',
metrics=[dice_coefficient])
# Display the model summary
model4.summary()
```

```
Model: "model_2"
__________________________________________________________________________
_______________________
 Layer (type)                Output Shape              Param #
Connected to
==========================================================================
===========================
 input_3 (InputLayer)        [(None, 128, 128, 3)]     0         []



 conv2d_66 (Conv2D)          (None, 64, 64, 64)        1792
['input_3[0][0]']


 batch_normalization_51 (Ba  (None, 64, 64, 64)        256
['conv2d_66[0][0]']
 tchNormalization)
```

```
 conv2d_67 (Conv2D)          (None, 64, 64, 64)           36928
['batch_normalization_51[0][0]
                                                                 ']



 batch_normalization_52 (Ba  (None, 64, 64, 64)           256
['conv2d_67[0][0]']
 tchNormalization)



 conv2d_68 (Conv2D)          (None, 64, 64, 64)           36928
['batch_normalization_52[0][0]
                                                                 ']



 conv2d_69 (Conv2D)          (None, 64, 64, 64)           4160
['batch_normalization_51[0][0]
                                                                 ']



 batch_normalization_53 (Ba  (None, 64, 64, 64)           256
['conv2d_68[0][0]']
 tchNormalization)



 batch_normalization_54 (Ba  (None, 64, 64, 64)           256
['conv2d_69[0][0]']
 tchNormalization)



 add_9 (Add)                 (None, 64, 64, 64)           0
['batch_normalization_53[0][0]
                                                                 ',


'batch_normalization_54[0][0]
                                                                 ']



 activation_33 (Activation)  (None, 64, 64, 64)           0
['add_9[0][0]']
```

```
 conv2d_70 (Conv2D)          (None, 32, 32, 128)          73856
['activation_33[0][0]']


 batch_normalization_55 (Ba  (None, 32, 32, 128)          512
['conv2d_70[0][0]']
 tchNormalization)



 conv2d_71 (Conv2D)          (None, 32, 32, 128)          147584
['batch_normalization_55[0][0]
                                                                  ']



 conv2d_72 (Conv2D)          (None, 32, 32, 128)          8320
['activation_33[0][0]']


 batch_normalization_56 (Ba  (None, 32, 32, 128)          512
['conv2d_71[0][0]']
 tchNormalization)



 batch_normalization_57 (Ba  (None, 32, 32, 128)          512
['conv2d_72[0][0]']
 tchNormalization)



 add_10 (Add)                (None, 32, 32, 128)          0
['batch_normalization_56[0][0]
                                                                  ',


'batch_normalization_57[0][0]
                                                                  ']



 activation_34 (Activation)  (None, 32, 32, 128)          0
['add_10[0][0]']


 conv2d_73 (Conv2D)          (None, 32, 32, 128)          147584
['activation_34[0][0]']


 batch_normalization_58 (Ba  (None, 32, 32, 128)          512
```

```
['conv2d_73[0][0]']
 tchNormalization)



 conv2d_74 (Conv2D)          (None, 32, 32, 128)          147584
['batch_normalization_58[0][0]
                                                                 ']



 conv2d_75 (Conv2D)          (None, 32, 32, 128)          16512
['activation_34[0][0]']


 batch_normalization_59 (Ba  (None, 32, 32, 128)          512
['conv2d_74[0][0]']
 tchNormalization)



 batch_normalization_60 (Ba  (None, 32, 32, 128)          512
['conv2d_75[0][0]']
 tchNormalization)



 add_11 (Add)                (None, 32, 32, 128)          0
['batch_normalization_59[0][0]
                                                                 ',


'batch_normalization_60[0][0]
                                                                 ']



 activation_35 (Activation)  (None, 32, 32, 128)          0
['add_11[0][0]']


 conv2d_76 (Conv2D)          (None, 16, 16, 256)          295168
['activation_35[0][0]']


 batch_normalization_61 (Ba  (None, 16, 16, 256)          1024
['conv2d_76[0][0]']
 tchNormalization)
```

```
 conv2d_77 (Conv2D)          (None, 16, 16, 256)          590080
['batch_normalization_61[0][0]
                                                                 ']



 conv2d_78 (Conv2D)          (None, 16, 16, 256)          33024
['activation_35[0][0]']


 batch_normalization_62 (Ba  (None, 16, 16, 256)          1024
['conv2d_77[0][0]']
 tchNormalization)



 batch_normalization_63 (Ba  (None, 16, 16, 256)          1024
['conv2d_78[0][0]']
 tchNormalization)



 add_12 (Add)                (None, 16, 16, 256)          0
['batch_normalization_62[0][0]
                                                                 ',


'batch_normalization_63[0][0]
                                                                 ']



 activation_36 (Activation)  (None, 16, 16, 256)          0
['add_12[0][0]']


 conv2d_79 (Conv2D)          (None, 16, 16, 256)          590080
['activation_36[0][0]']


 batch_normalization_64 (Ba  (None, 16, 16, 256)          1024
['conv2d_79[0][0]']
 tchNormalization)



 conv2d_80 (Conv2D)          (None, 16, 16, 256)          590080
['batch_normalization_64[0][0]
                                                                 ']
```

```
 conv2d_81 (Conv2D)          (None, 16, 16, 256)          65792
['activation_36[0][0]']


 batch_normalization_65 (Ba  (None, 16, 16, 256)          1024
['conv2d_80[0][0]']
 tchNormalization)



 batch_normalization_66 (Ba  (None, 16, 16, 256)          1024
['conv2d_81[0][0]']
 tchNormalization)



 add_13 (Add)                (None, 16, 16, 256)          0
['batch_normalization_65[0][0]
                                                                  ',


'batch_normalization_66[0][0]
                                                                  ']



 activation_37 (Activation)  (None, 16, 16, 256)          0
['add_13[0][0]']


 conv2d_82 (Conv2D)          (None, 8, 8, 512)            1180160
['activation_37[0][0]']


 batch_normalization_67 (Ba  (None, 8, 8, 512)            2048
['conv2d_82[0][0]']
 tchNormalization)



 conv2d_83 (Conv2D)          (None, 8, 8, 512)            2359808
['batch_normalization_67[0][0]
                                                                  ']



 conv2d_84 (Conv2D)          (None, 8, 8, 512)            131584
['activation_37[0][0]']
```

```
 batch_normalization_68 (Ba  (None, 8, 8, 512)            2048
['conv2d_83[0][0]']
 tchNormalization)



 batch_normalization_69 (Ba  (None, 8, 8, 512)            2048
['conv2d_84[0][0]']
 tchNormalization)



 add_14 (Add)                (None, 8, 8, 512)            0
['batch_normalization_68[0][0]
                                                                  ',


'batch_normalization_69[0][0]
                                                                  ']



 activation_38 (Activation)  (None, 8, 8, 512)            0
['add_14[0][0]']


 global_average_pooling2d (  (None, 512)                  0
['activation_38[0][0]']
 GlobalAveragePooling2D)



 reshape (Reshape)           (None, 1, 1, 512)            0
['global_average_pooling2d[0][

0]']


 up_sampling2d_12 (UpSampli  (None, 8, 8, 512)            0
['reshape[0][0]']
 ng2D)



 conv2d_85 (Conv2D)          (None, 8, 8, 512)            262656
['up_sampling2d_12[0][0]']


 conv2d_86 (Conv2D)          (None, 8, 8, 512)            262656
['activation_38[0][0]']
```

```
 concatenate_6 (Concatenate  (None, 8, 8, 1024)           0
['conv2d_85[0][0]',
 )
'conv2d_86[0][0]']


 conv2d_87 (Conv2D)          (None, 8, 8, 512)            4719104
['concatenate_6[0][0]']


 batch_normalization_70 (Ba  (None, 8, 8, 512)            2048
['conv2d_87[0][0]']
 tchNormalization)



 up_sampling2d_13 (UpSampli  (None, 16, 16, 512)          0
['batch_normalization_70[0][0]
 ng2D)                                                                 ']



 conv2d_88 (Conv2D)          (None, 16, 16, 256)          1179904
['up_sampling2d_13[0][0]']


 batch_normalization_71 (Ba  (None, 16, 16, 256)          1024
['conv2d_88[0][0]']
 tchNormalization)



 up_sampling2d_14 (UpSampli  (None, 32, 32, 256)          0
['batch_normalization_71[0][0]
 ng2D)                                                                 ']



 conv2d_89 (Conv2D)          (None, 32, 32, 128)          295040
['up_sampling2d_14[0][0]']


 batch_normalization_72 (Ba  (None, 32, 32, 128)          512
['conv2d_89[0][0]']
 tchNormalization)



 up_sampling2d_15 (UpSampli  (None, 64, 64, 128)          0
['batch_normalization_72[0][0]
```

```
 ng2D)                                                                  ']



 conv2d_90 (Conv2D)          (None, 64, 64, 64)           73792
['up_sampling2d_15[0][0]']


 batch_normalization_73 (Ba  (None, 64, 64, 64)           256
['conv2d_90[0][0]']
 tchNormalization)



 up_sampling2d_16 (UpSampli  (None, 128, 128, 64)         0
['batch_normalization_73[0][0]
 ng2D)                                                                  ']



 conv2d_91 (Conv2D)          (None, 128, 128, 1)          65
['up_sampling2d_16[0][0]']


================================================================================
Total params: 13270465 (50.62 MB)
Trainable params: 13260353 (50.58 MB)
Non-trainable params: 10112 (39.50 KB)
________________________________________________________________________________
```

```
model4.compile(optimizer=Adam(lr=0.001, beta_1=0.9,
beta_2=0.999),loss= 'binary_crossentropy', metrics=[dice_coefficient])
model4.save('Pspnet_model4.h5')

checkpoint = tf.keras.callbacks.ModelCheckpoint("Pspnet_model4.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                   patience=5,
                                                   mode='auto',

restore_best_weights=True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                  factor=0.2,
                                                  patience=2,
                                                  min_lr=1e-6,
                                                  mode='auto',
```

```
                                                    verbose=1)
# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]

history4= model4.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model4.save('Pspnet_model4.h5')
```

```
Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.2047 -
dice_coefficient: 0.8819
Epoch 1: val_loss improved from inf to 0.08324, saving model to
Pspnet_model4.h5
394/394 [==============================] - 41s 60ms/step - loss:
0.2047 - dice_coefficient: 0.8819 - val_loss: 0.0832 -
val_dice_coefficient: 0.9770 - lr: 0.0010
Epoch 2/10
394/394 [==============================] - ETA: 0s - loss: 0.0333 -
dice_coefficient: 0.9843
Epoch 2: val_loss improved from 0.08324 to 0.03723, saving model to
Pspnet_model4.h5
394/394 [==============================] - 21s 54ms/step - loss:
0.0333 - dice_coefficient: 0.9843 - val_loss: 0.0372 -
val_dice_coefficient: 0.9799 - lr: 0.0010
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: 0.0228 -
dice_coefficient: 0.9905
Epoch 3: val_loss did not improve from 0.03723
394/394 [==============================] - 21s 52ms/step - loss:
0.0228 - dice_coefficient: 0.9905 - val_loss: 0.1126 -
val_dice_coefficient: 0.9882 - lr: 0.0010
Epoch 4/10
394/394 [==============================] - ETA: 0s - loss: 0.0246 -
dice_coefficient: 0.9908
Epoch 4: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.

Epoch 4: val_loss did not improve from 0.03723
394/394 [==============================] - 20s 52ms/step - loss:
0.0246 - dice_coefficient: 0.9908 - val_loss: 0.0677 -
val_dice_coefficient: 0.9897 - lr: 0.0010
Epoch 5/10
393/394 [==============================>.] - ETA: 0s - loss: 0.0054 -
dice_coefficient: 0.9907
Epoch 5: val_loss improved from 0.03723 to 0.02111, saving model to
Pspnet_model4.h5
394/394 [==============================] - 21s 53ms/step - loss:
0.0054 - dice_coefficient: 0.9907 - val_loss: 0.0211 -
val_dice_coefficient: 0.9923 - lr: 2.0000e-04
```

```
Epoch 6/10
394/394 [==============================] - ETA: 0s - loss: -0.0061 -
dice_coefficient: 0.9893
Epoch 6: val_loss did not improve from 0.02111
394/394 [==============================] - 21s 52ms/step - loss: -
0.0061 - dice_coefficient: 0.9893 - val_loss: 0.1229 -
val_dice_coefficient: 0.9851 - lr: 2.0000e-04
Epoch 7/10
393/394 [============================>.] - ETA: 0s - loss: -0.0058 -
dice_coefficient: 0.9894
Epoch 7: val_loss improved from 0.02111 to -0.16824, saving model to
Pspnet_model4.h5
394/394 [==============================] - 21s 53ms/step - loss: -
0.0057 - dice_coefficient: 0.9894 - val_loss: -0.1682 -
val_dice_coefficient: 0.9894 - lr: 2.0000e-04
Epoch 8/10
394/394 [==============================] - ETA: 0s - loss: -0.0567 -
dice_coefficient: 0.9923
Epoch 8: val_loss did not improve from -0.16824
394/394 [==============================] - 20s 52ms/step - loss: -
0.0567 - dice_coefficient: 0.9923 - val_loss: 0.1876 -
val_dice_coefficient: 0.9922 - lr: 2.0000e-04
Epoch 9/10
393/394 [============================>.] - ETA: 0s - loss: -0.0466 -
dice_coefficient: 0.9914
Epoch 9: ReduceLROnPlateau reducing learning rate to
4.0000001899898055e-05.

Epoch 9: val_loss did not improve from -0.16824
394/394 [==============================] - 20s 52ms/step - loss: -
0.0464 - dice_coefficient: 0.9914 - val_loss: -0.0190 -
val_dice_coefficient: 0.9925 - lr: 2.0000e-04
Epoch 10/10
393/394 [============================>.] - ETA: 0s - loss: -0.0747 -
dice_coefficient: 0.9920
Epoch 10: val_loss did not improve from -0.16824
394/394 [==============================] - 20s 52ms/step - loss: -
0.0744 - dice_coefficient: 0.9920 - val_loss: -0.0265 -
val_dice_coefficient: 0.9939 - lr: 4.0000e-05

scores4 = model4.evaluate(x_valid, y_valid)
scores4[1]

57/57 [==============================] - 2s 26ms/step - loss: -0.0265
- dice_coefficient: 0.9939

0.99385005235672
```

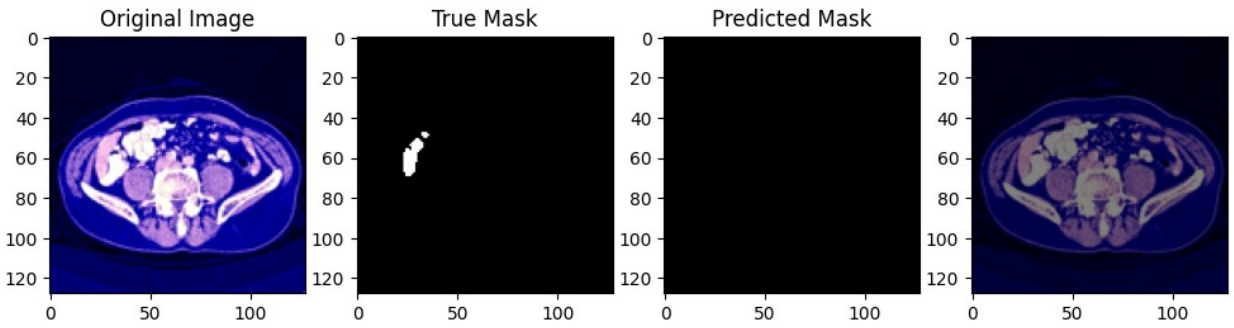
```
prediction4 = model4.predict(x_test)
test_scores4 = model4.evaluate(x_test, y_test)
test_scores4[1]
```

```
29/29 [==============================] - 1s 31ms/step
29/29 [==============================] - 1s 26ms/step - loss: 1.3909 -
dice_coefficient: 0.9934
```

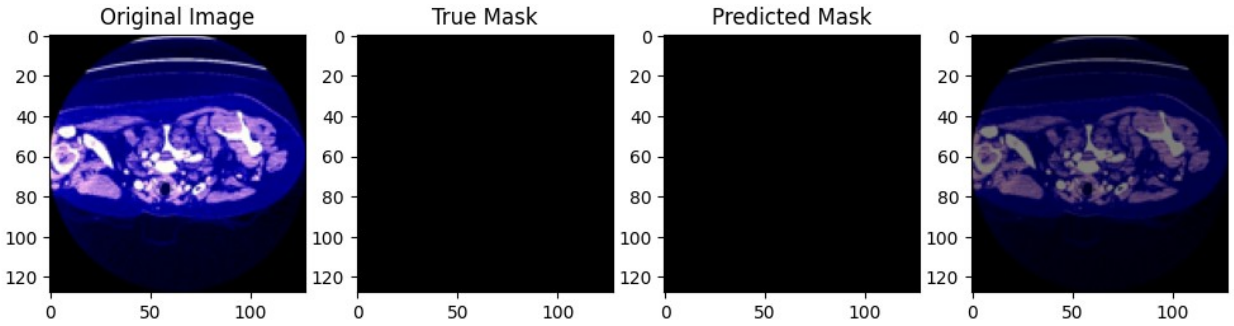
```
0.9934154152870178
```

```
import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 20, replace=False)
for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model4
    predicted_mask = model4.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
    threshold = 0.5  # Adjust this threshold based on your model's
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')
    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
```

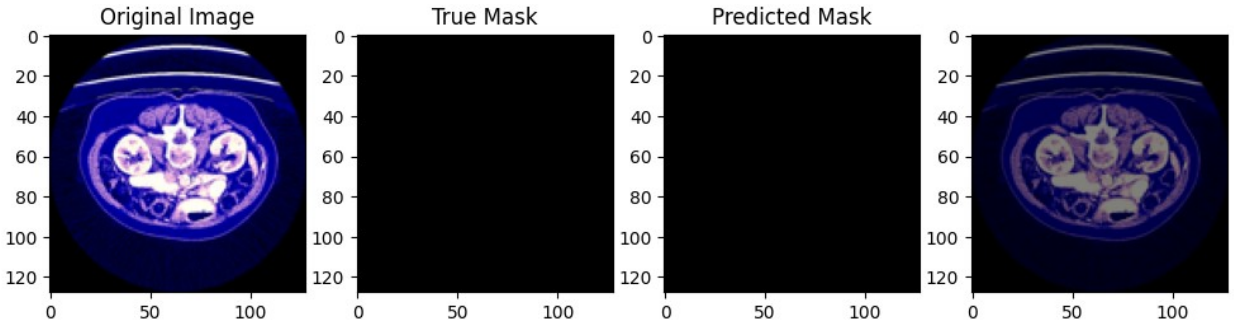
```
'nipy_spectral')
    plt.show()

333
1/1 [==============================] - 0s 191ms/step
```



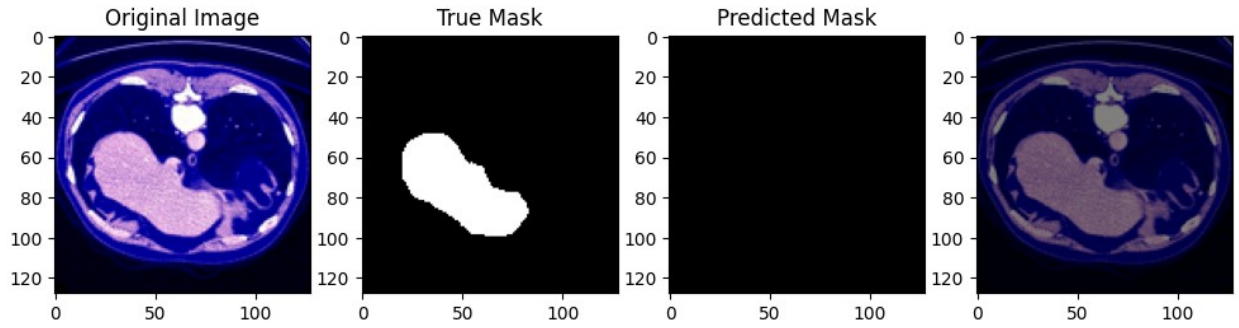

```
812
1/1 [==============================] - 0s 24ms/step
```



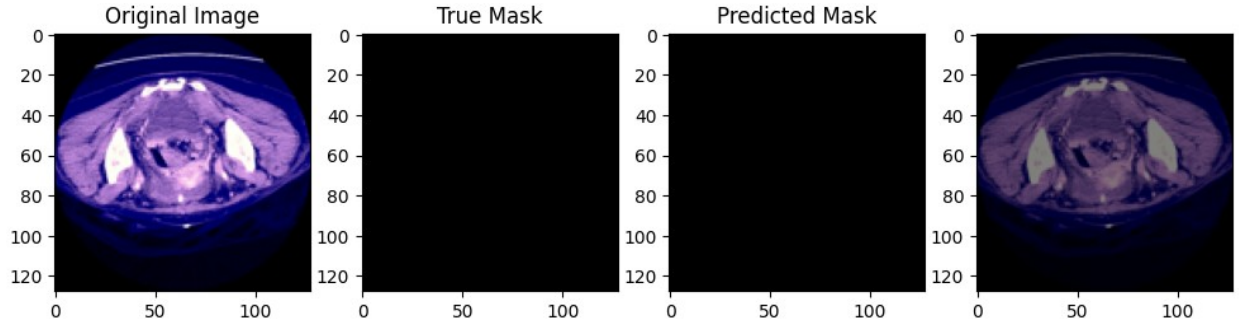

```
745
1/1 [==============================] - 0s 24ms/step
```



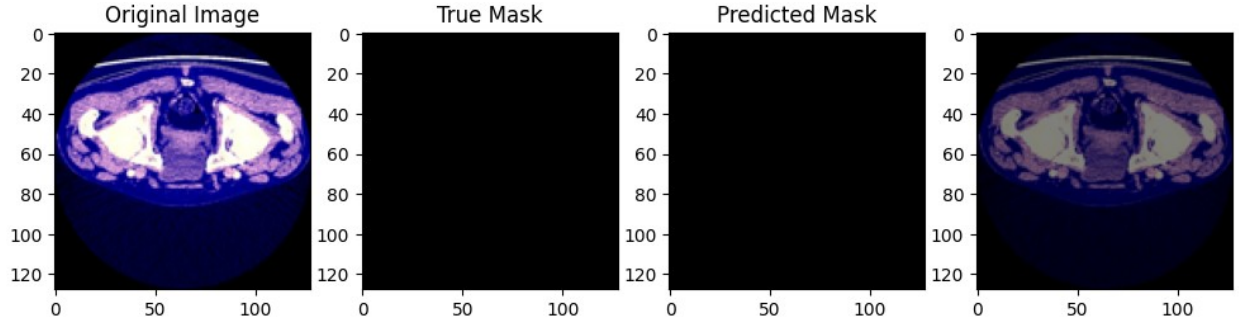

```
245
1/1 [==============================] - 0s 22ms/step
```



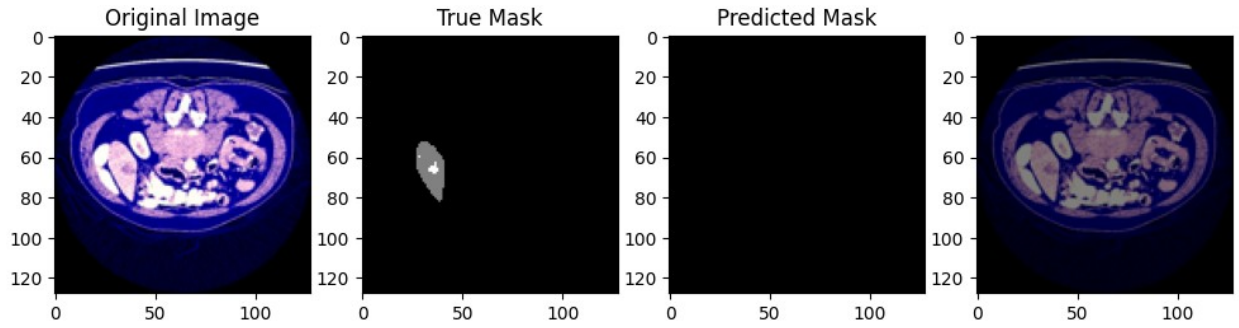

```
40
1/1 [==============================] - 0s 23ms/step
```



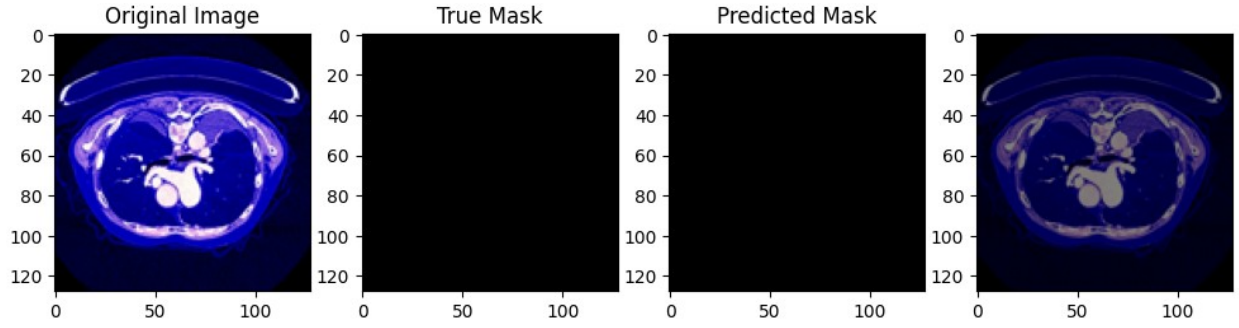

```
427
1/1 [==============================] - 0s 22ms/step
```



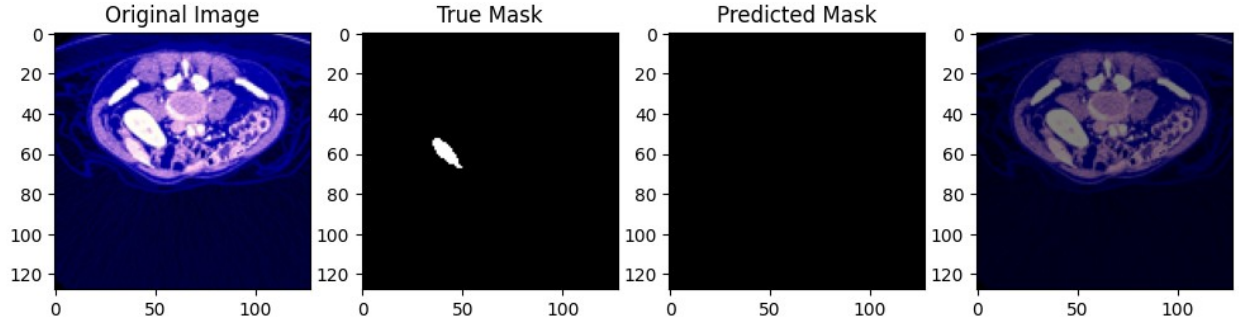

```
303
1/1 [==============================] - 0s 23ms/step
```



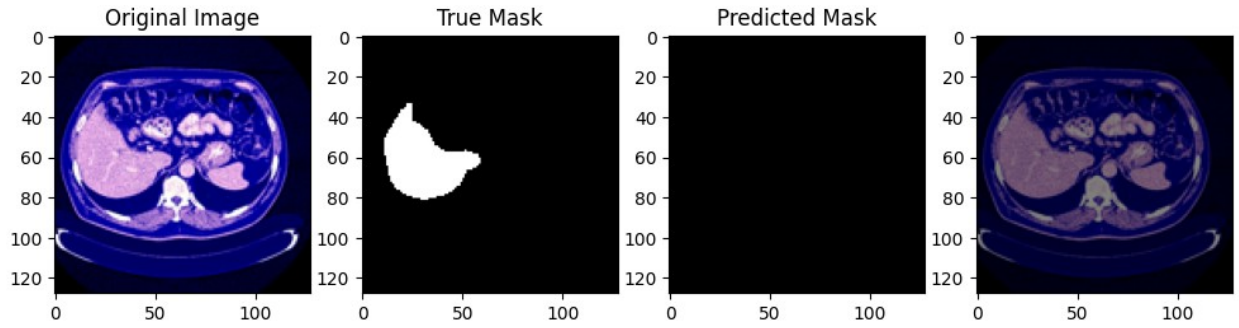

```
588
1/1 [==============================] - 0s 24ms/step
```



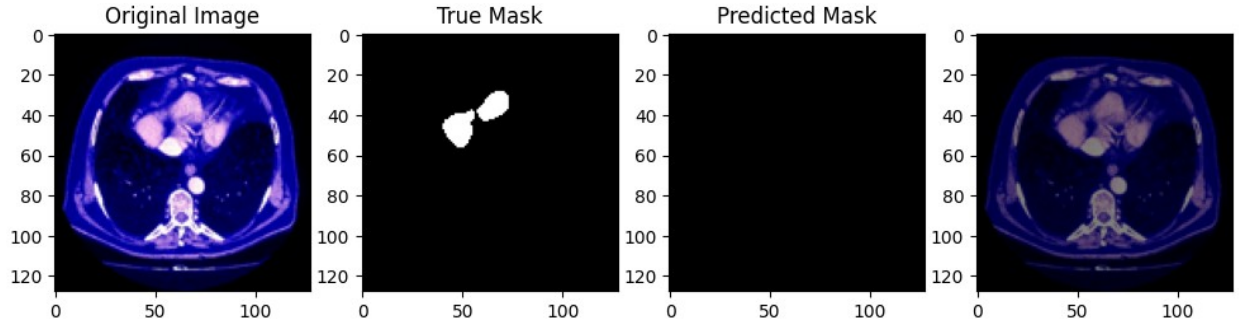

```
210
1/1 [==============================] - 0s 22ms/step
```



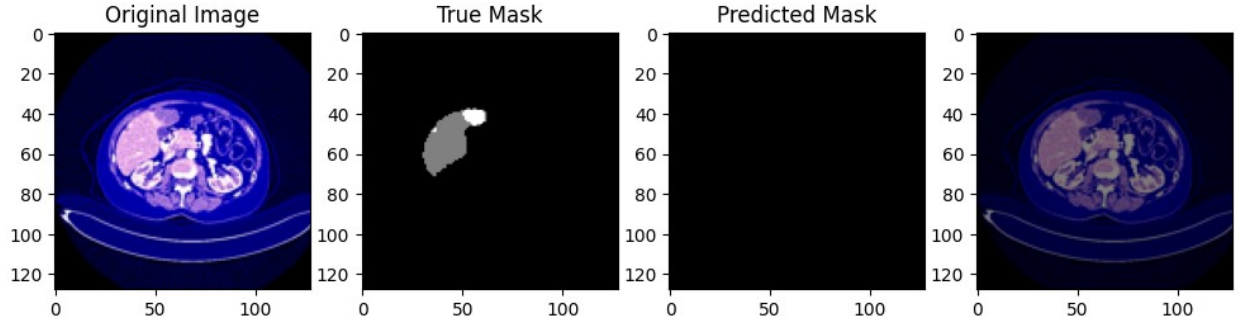

```
137
1/1 [==============================] - 0s 22ms/step
```



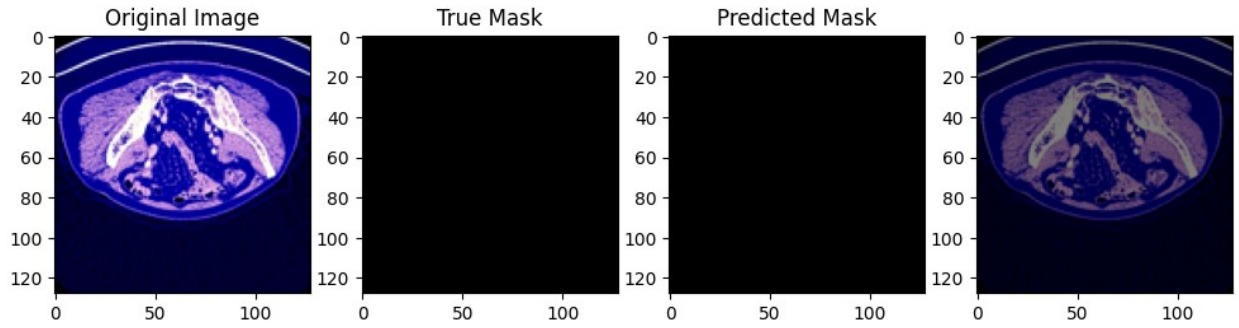

```
138
1/1 [==============================] - 0s 22ms/step
```



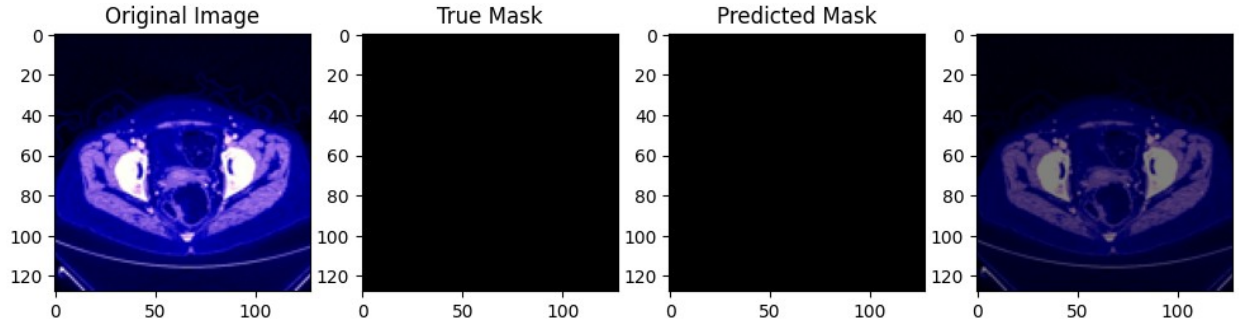

```
261
1/1 [==============================] - 0s 28ms/step
```



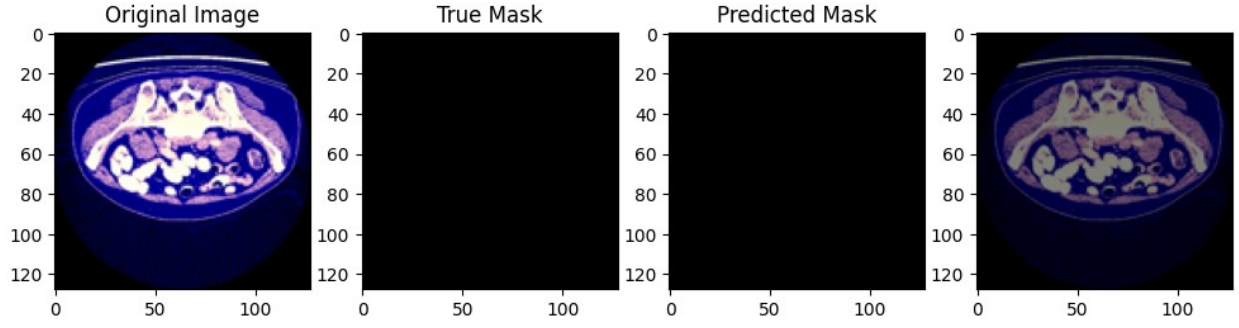

```
352
1/1 [==============================] - 0s 22ms/step
```



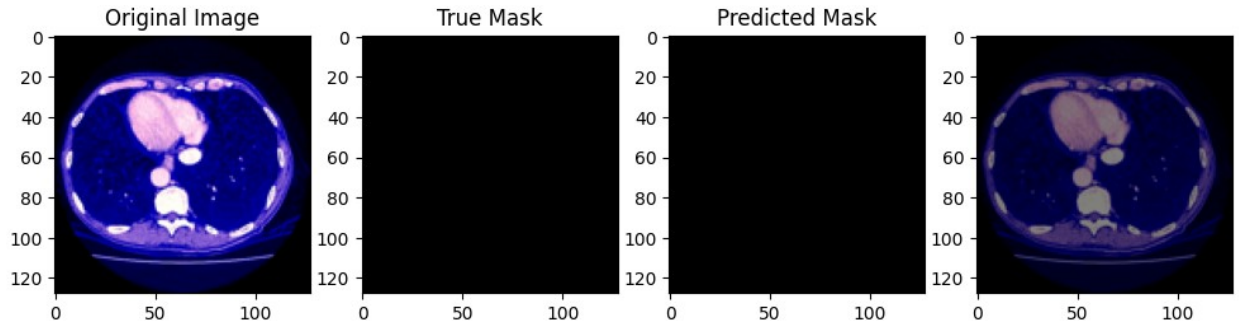

```
360
1/1 [==============================] - 0s 23ms/step
```



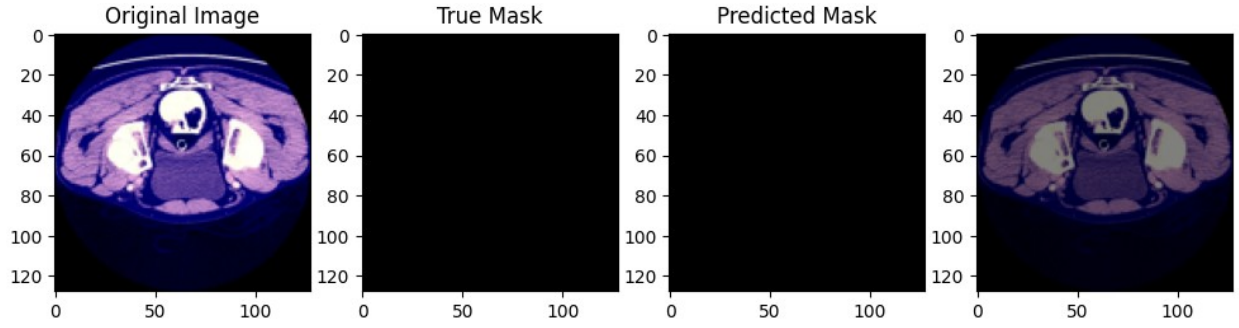

```
347
1/1 [==============================] - 0s 23ms/step
```



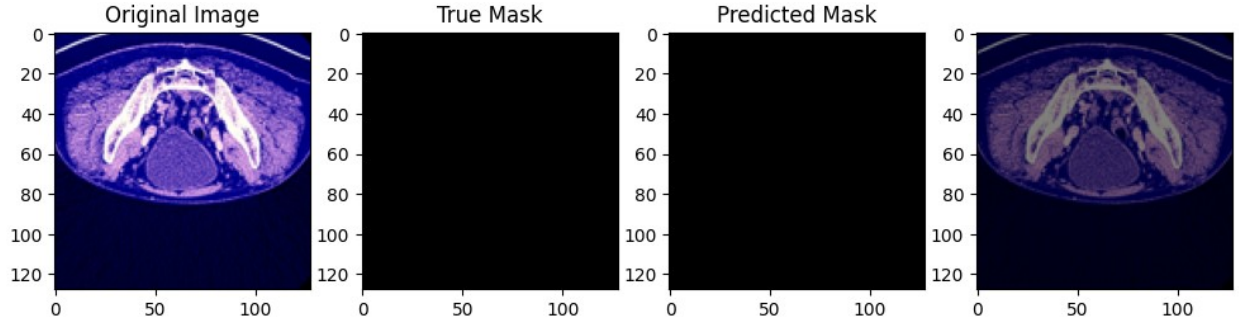

```
479
1/1 [==============================] - 0s 22ms/step
```



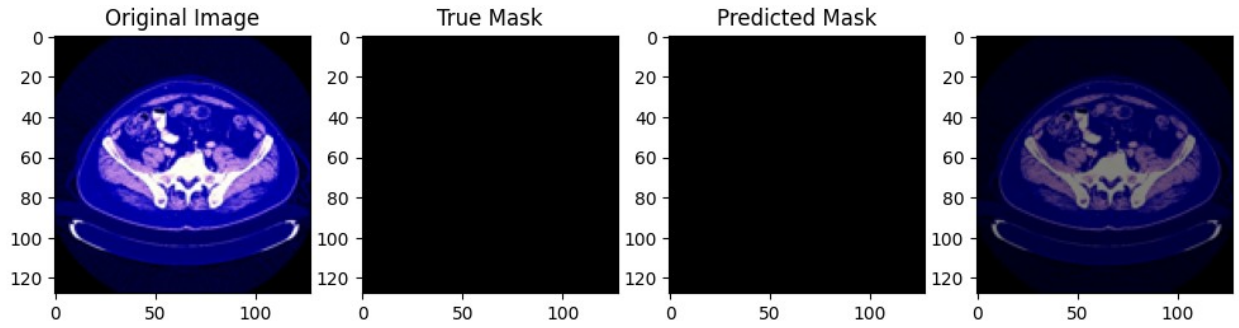

```
307
1/1 [==============================] - 0s 26ms/step
```



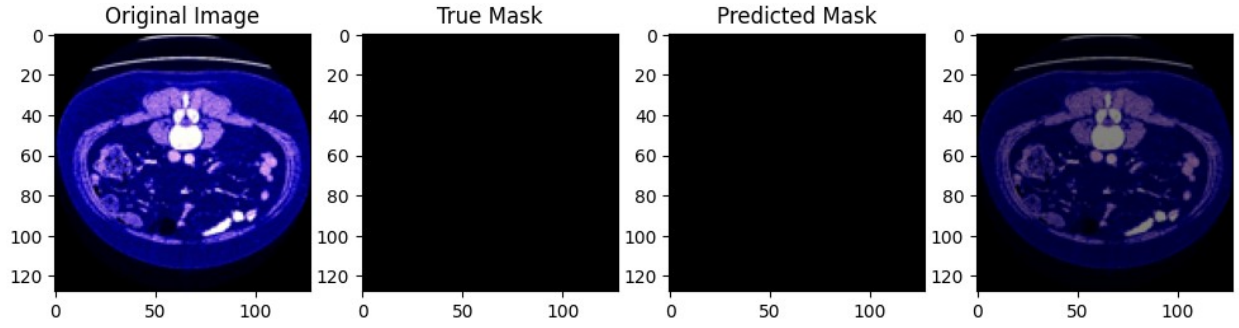

```
658
1/1 [==============================] - 0s 23ms/step
```



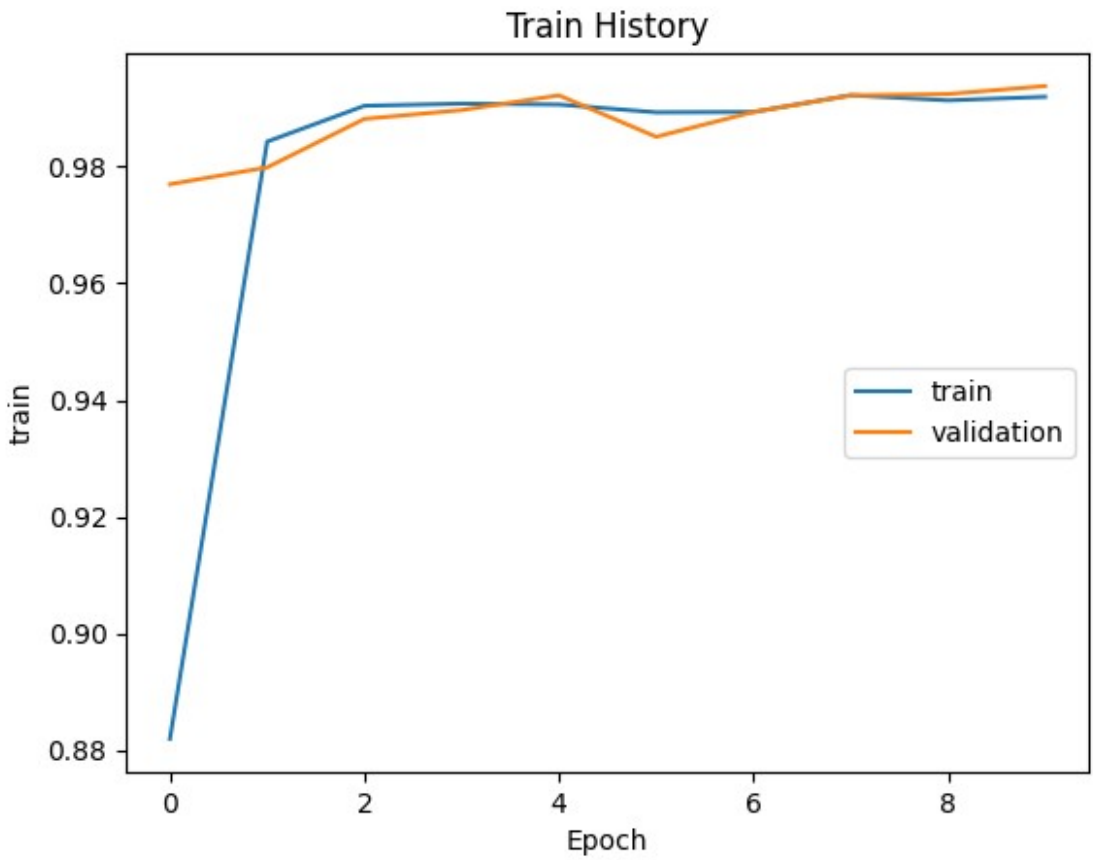

```
895
1/1 [==============================] - 0s 23ms/step
```



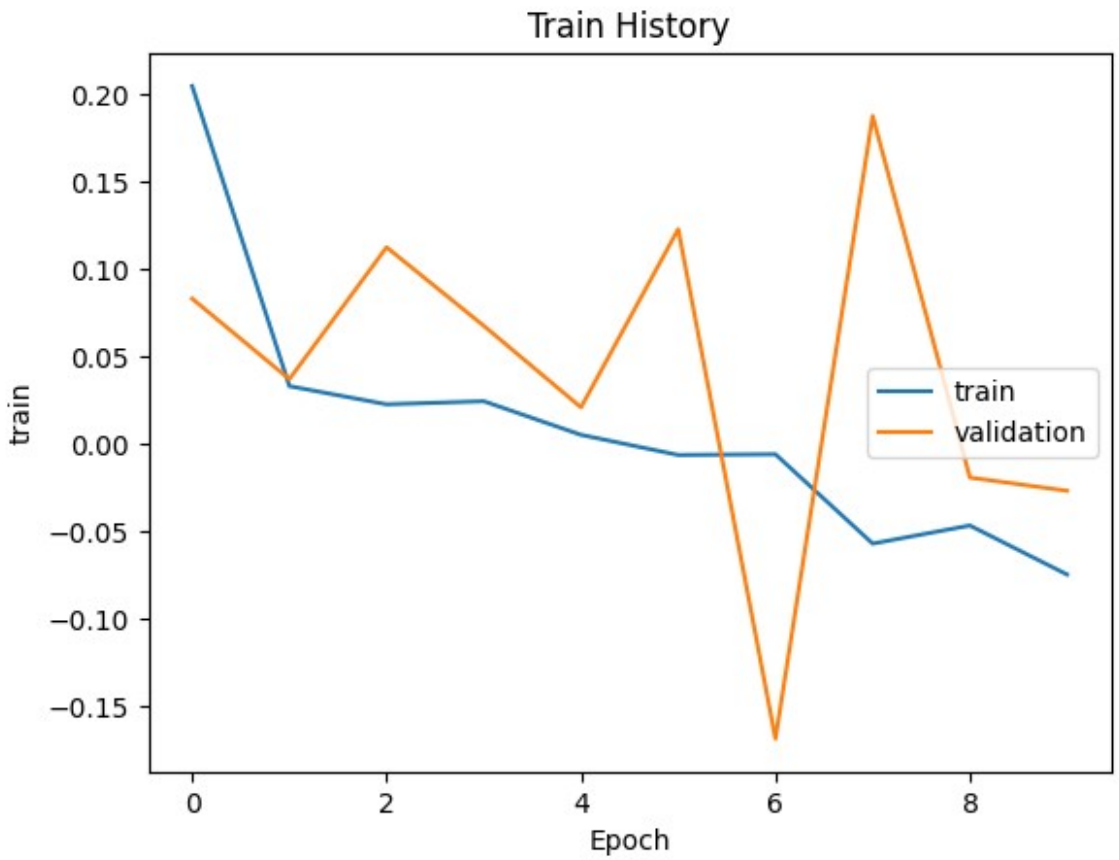

```
111
1/1 [==============================] - 0s 23ms/step
```




```
show_history(history4, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history4, 'loss', 'val_loss')
```




```
import tensorflow as tf
from tensorflow import keras

def dense_block(x, filters, blocks):
    for i in range(blocks):
        x = conv_block(x, filters)
    return x

def transition_block(x, reduction):
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(int(keras.backend.int_shape(x)[-1] *
reduction), 1, use_bias=False)(x)
    x = keras.layers.Activation('relu')(x)
    x = keras.layers.AveragePooling2D(2)(x)
    return x

def conv_block(x, growth_rate):
    x1 = keras.layers.BatchNormalization()(x)
    x1 = keras.layers.Activation('relu')(x1)
    x1 = keras.layers.Conv2D(growth_rate, 3, padding='same',
use_bias=False)(x1)
    x = keras.layers.concatenate([x, x1], axis=-1)
    return x
```

```
def DenseNet():
    f = 32  # Growth rate
    inputs = keras.layers.Input((128, 128, 3))

    # Initial convolution layer
    x = keras.layers.Conv2D(64, 7, strides=2, padding='same',
use_bias=False)(inputs)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Activation('relu')(x)
    x = keras.layers.MaxPooling2D(3, strides=2, padding='same')(x)

    # Dense blocks
    x = dense_block(x, f, 6)
    x = transition_block(x, 0.5)
    x = dense_block(x, f, 12)
    x = transition_block(x, 0.5)
    x = dense_block(x, f, 24)
    x = transition_block(x, 0.5)
    x = dense_block(x, f, 16)  # Adjusted number of blocks to match
output size

    # Final layers
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Activation('relu')(x)
    x = keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(x)
    outputs = keras.layers.UpSampling2D((32, 32))(x)  # Adjusted
upsampling factor to match label size

    model = keras.models.Model(inputs, outputs)
    return model

# Instantiate the model
model5 = DenseNet()

# Compile the model
model5.compile(optimizer='Adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Display the model summary
model5.summary()
```

```
Model: "model_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
Connected to
=================================================================
 input_12 (InputLayer)       [(None, 128, 128, 3)]     0         []
```

```
 conv2d_596 (Conv2D)         (None, 64, 64, 64)           9408
['input_12[0][0]']


 batch_normalization_578 (B  (None, 64, 64, 64)           256
['conv2d_596[0][0]']
 atchNormalization)



 activation_543 (Activation  (None, 64, 64, 64)           0
['batch_normalization_578[0][0
 )                                                                ]'
]


 max_pooling2d_16 (MaxPooli  (None, 32, 32, 64)           0
['activation_543[0][0]']
 ng2D)



 batch_normalization_579 (B  (None, 32, 32, 64)           256
['max_pooling2d_16[0][0]']
 atchNormalization)



 activation_544 (Activation  (None, 32, 32, 64)           0
['batch_normalization_579[0][0
 )                                                                ]'
]


 conv2d_597 (Conv2D)         (None, 32, 32, 32)           18432
['activation_544[0][0]']


 concatenate_471 (Concatena  (None, 32, 32, 96)           0
['max_pooling2d_16[0][0]',
 te)
'conv2d_597[0][0]']


 batch_normalization_580 (B  (None, 32, 32, 96)           384
['concatenate_471[0][0]']
 atchNormalization)
```

```
 activation_545 (Activation  (None, 32, 32, 96)           0
['batch_normalization_580[0][0
 )                                                                ]'
]


 conv2d_598 (Conv2D)         (None, 32, 32, 32)           27648
['activation_545[0][0]']


 concatenate_472 (Concatena  (None, 32, 32, 128)          0
['concatenate_471[0][0]',
 te)
'conv2d_598[0][0]']


 batch_normalization_581 (B  (None, 32, 32, 128)          512
['concatenate_472[0][0]']
 atchNormalization)



 activation_546 (Activation  (None, 32, 32, 128)          0
['batch_normalization_581[0][0
 )                                                                ]'
]


 conv2d_599 (Conv2D)         (None, 32, 32, 32)           36864
['activation_546[0][0]']


 concatenate_473 (Concatena  (None, 32, 32, 160)          0
['concatenate_472[0][0]',
 te)
'conv2d_599[0][0]']


 batch_normalization_582 (B  (None, 32, 32, 160)          640
['concatenate_473[0][0]']
 atchNormalization)



 activation_547 (Activation  (None, 32, 32, 160)          0
['batch_normalization_582[0][0
 )                                                                ]'
]
```

```
 conv2d_600 (Conv2D)         (None, 32, 32, 32)           46080
['activation_547[0][0]']


 concatenate_474 (Concatena  (None, 32, 32, 192)          0
['concatenate_473[0][0]',
 te)
'conv2d_600[0][0]']


 batch_normalization_583 (B  (None, 32, 32, 192)          768
['concatenate_474[0][0]']
 atchNormalization)



 activation_548 (Activation  (None, 32, 32, 192)          0
['batch_normalization_583[0][0
 )                                                                ]'
]


 conv2d_601 (Conv2D)         (None, 32, 32, 32)           55296
['activation_548[0][0]']


 concatenate_475 (Concatena  (None, 32, 32, 224)          0
['concatenate_474[0][0]',
 te)
'conv2d_601[0][0]']


 batch_normalization_584 (B  (None, 32, 32, 224)          896
['concatenate_475[0][0]']
 atchNormalization)



 activation_549 (Activation  (None, 32, 32, 224)          0
['batch_normalization_584[0][0
 )                                                                ]'
]


 conv2d_602 (Conv2D)         (None, 32, 32, 32)           64512
['activation_549[0][0]']
```

```
 concatenate_476 (Concatena  (None, 32, 32, 256)          0
['concatenate_475[0][0]',
 te)
'conv2d_602[0][0]']


 batch_normalization_585 (B  (None, 32, 32, 256)          1024
['concatenate_476[0][0]']
 atchNormalization)



 conv2d_603 (Conv2D)         (None, 32, 32, 128)          32768
['batch_normalization_585[0][0
                                                                  ]'
]


 activation_550 (Activation  (None, 32, 32, 128)          0
['conv2d_603[0][0]']
 )



 average_pooling2d_24 (Aver  (None, 16, 16, 128)          0
['activation_550[0][0]']
 agePooling2D)



 batch_normalization_586 (B  (None, 16, 16, 128)          512
['average_pooling2d_24[0][0]']
 atchNormalization)



 activation_551 (Activation  (None, 16, 16, 128)          0
['batch_normalization_586[0][0
 )                                                                ]'
]


 conv2d_604 (Conv2D)         (None, 16, 16, 32)           36864
['activation_551[0][0]']


 concatenate_477 (Concatena  (None, 16, 16, 160)          0
['average_pooling2d_24[0][0]',
 te)
'conv2d_604[0][0]']
```

```
 batch_normalization_587 (B  (None, 16, 16, 160)          640
['concatenate_477[0][0]']
 atchNormalization)



 activation_552 (Activation  (None, 16, 16, 160)          0
['batch_normalization_587[0][0
 )                                                                ]'
]


 conv2d_605 (Conv2D)         (None, 16, 16, 32)           46080
['activation_552[0][0]']


 concatenate_478 (Concatena  (None, 16, 16, 192)          0
['concatenate_477[0][0]',
 te)
'conv2d_605[0][0]']


 batch_normalization_588 (B  (None, 16, 16, 192)          768
['concatenate_478[0][0]']
 atchNormalization)



 activation_553 (Activation  (None, 16, 16, 192)          0
['batch_normalization_588[0][0
 )                                                                ]'
]


 conv2d_606 (Conv2D)         (None, 16, 16, 32)           55296
['activation_553[0][0]']


 concatenate_479 (Concatena  (None, 16, 16, 224)          0
['concatenate_478[0][0]',
 te)
'conv2d_606[0][0]']


 batch_normalization_589 (B  (None, 16, 16, 224)          896
['concatenate_479[0][0]']
 atchNormalization)
```

```
 activation_554 (Activation  (None, 16, 16, 224)          0
['batch_normalization_589[0][0
 )                                                                ]'
]


 conv2d_607 (Conv2D)         (None, 16, 16, 32)           64512
['activation_554[0][0]']


 concatenate_480 (Concatena  (None, 16, 16, 256)          0
['concatenate_479[0][0]',
 te)
'conv2d_607[0][0]']


 batch_normalization_590 (B  (None, 16, 16, 256)          1024
['concatenate_480[0][0]']
 atchNormalization)



 activation_555 (Activation  (None, 16, 16, 256)          0
['batch_normalization_590[0][0
 )                                                                ]'
]


 conv2d_608 (Conv2D)         (None, 16, 16, 32)           73728
['activation_555[0][0]']


 concatenate_481 (Concatena  (None, 16, 16, 288)          0
['concatenate_480[0][0]',
 te)
'conv2d_608[0][0]']


 batch_normalization_591 (B  (None, 16, 16, 288)          1152
['concatenate_481[0][0]']
 atchNormalization)



 activation_556 (Activation  (None, 16, 16, 288)          0
['batch_normalization_591[0][0
 )                                                                ]'
]
```

```
 conv2d_609 (Conv2D)         (None, 16, 16, 32)           82944
['activation_556[0][0]']


 concatenate_482 (Concatena  (None, 16, 16, 320)          0
['concatenate_481[0][0]',
 te)
'conv2d_609[0][0]']


 batch_normalization_592 (B  (None, 16, 16, 320)          1280
['concatenate_482[0][0]']
 atchNormalization)



 activation_557 (Activation  (None, 16, 16, 320)          0
['batch_normalization_592[0][0
 )                                                                ]'
]


 conv2d_610 (Conv2D)         (None, 16, 16, 32)           92160
['activation_557[0][0]']


 concatenate_483 (Concatena  (None, 16, 16, 352)          0
['concatenate_482[0][0]',
 te)
'conv2d_610[0][0]']


 batch_normalization_593 (B  (None, 16, 16, 352)          1408
['concatenate_483[0][0]']
 atchNormalization)



 activation_558 (Activation  (None, 16, 16, 352)          0
['batch_normalization_593[0][0
 )                                                                ]'
]


 conv2d_611 (Conv2D)         (None, 16, 16, 32)           101376
['activation_558[0][0]']
```

```
 concatenate_484 (Concatena  (None, 16, 16, 384)          0
['concatenate_483[0][0]',
 te)
'conv2d_611[0][0]']


 batch_normalization_594 (B  (None, 16, 16, 384)          1536
['concatenate_484[0][0]']
 atchNormalization)



 activation_559 (Activation  (None, 16, 16, 384)          0
['batch_normalization_594[0][0
 )                                                                ]'
]


 conv2d_612 (Conv2D)         (None, 16, 16, 32)           110592
['activation_559[0][0]']


 concatenate_485 (Concatena  (None, 16, 16, 416)          0
['concatenate_484[0][0]',
 te)
'conv2d_612[0][0]']


 batch_normalization_595 (B  (None, 16, 16, 416)          1664
['concatenate_485[0][0]']
 atchNormalization)



 activation_560 (Activation  (None, 16, 16, 416)          0
['batch_normalization_595[0][0
 )                                                                ]'
]


 conv2d_613 (Conv2D)         (None, 16, 16, 32)           119808
['activation_560[0][0]']


 concatenate_486 (Concatena  (None, 16, 16, 448)          0
['concatenate_485[0][0]',
 te)
'conv2d_613[0][0]']
```

```
 batch_normalization_596 (B  (None, 16, 16, 448)          1792
['concatenate_486[0][0]']
 atchNormalization)



 activation_561 (Activation  (None, 16, 16, 448)          0
['batch_normalization_596[0][0
 )                                                                ]'
]


 conv2d_614 (Conv2D)         (None, 16, 16, 32)           129024
['activation_561[0][0]']


 concatenate_487 (Concatena  (None, 16, 16, 480)          0
['concatenate_486[0][0]',
 te)
'conv2d_614[0][0]']


 batch_normalization_597 (B  (None, 16, 16, 480)          1920
['concatenate_487[0][0]']
 atchNormalization)



 activation_562 (Activation  (None, 16, 16, 480)          0
['batch_normalization_597[0][0
 )                                                                ]'
]


 conv2d_615 (Conv2D)         (None, 16, 16, 32)           138240
['activation_562[0][0]']


 concatenate_488 (Concatena  (None, 16, 16, 512)          0
['concatenate_487[0][0]',
 te)
'conv2d_615[0][0]']


 batch_normalization_598 (B  (None, 16, 16, 512)          2048
['concatenate_488[0][0]']
 atchNormalization)
```

```
 conv2d_616 (Conv2D)         (None, 16, 16, 256)          131072
['batch_normalization_598[0][0
                                                                  ]'
]


 activation_563 (Activation  (None, 16, 16, 256)          0
['conv2d_616[0][0]']
 )



 average_pooling2d_25 (Aver  (None, 8, 8, 256)            0
['activation_563[0][0]']
 agePooling2D)



 batch_normalization_599 (B  (None, 8, 8, 256)            1024
['average_pooling2d_25[0][0]']
 atchNormalization)



 activation_564 (Activation  (None, 8, 8, 256)            0
['batch_normalization_599[0][0
 )                                                                ]'
]


 conv2d_617 (Conv2D)         (None, 8, 8, 32)             73728
['activation_564[0][0]']


 concatenate_489 (Concatena  (None, 8, 8, 288)            0
['average_pooling2d_25[0][0]',
 te)
'conv2d_617[0][0]']


 batch_normalization_600 (B  (None, 8, 8, 288)            1152
['concatenate_489[0][0]']
 atchNormalization)



 activation_565 (Activation  (None, 8, 8, 288)            0
['batch_normalization_600[0][0
 )                                                                ]'
]
```

```
 conv2d_618 (Conv2D)         (None, 8, 8, 32)             82944
['activation_565[0][0]']


 concatenate_490 (Concatena  (None, 8, 8, 320)            0
['concatenate_489[0][0]',
 te)
'conv2d_618[0][0]']


 batch_normalization_601 (B  (None, 8, 8, 320)            1280
['concatenate_490[0][0]']
 atchNormalization)



 activation_566 (Activation  (None, 8, 8, 320)            0
['batch_normalization_601[0][0
 )                                                                ]'
]


 conv2d_619 (Conv2D)         (None, 8, 8, 32)             92160
['activation_566[0][0]']


 concatenate_491 (Concatena  (None, 8, 8, 352)            0
['concatenate_490[0][0]',
 te)
'conv2d_619[0][0]']


 batch_normalization_602 (B  (None, 8, 8, 352)            1408
['concatenate_491[0][0]']
 atchNormalization)



 activation_567 (Activation  (None, 8, 8, 352)            0
['batch_normalization_602[0][0
 )                                                                ]'
]


 conv2d_620 (Conv2D)         (None, 8, 8, 32)             101376
['activation_567[0][0]']


 concatenate_492 (Concatena  (None, 8, 8, 384)            0
```

```
['concatenate_491[0][0]',
 te)
'conv2d_620[0][0]']


 batch_normalization_603 (B  (None, 8, 8, 384)            1536
['concatenate_492[0][0]']
 atchNormalization)



 activation_568 (Activation  (None, 8, 8, 384)            0
['batch_normalization_603[0][0
 )                                                                ]'
]


 conv2d_621 (Conv2D)         (None, 8, 8, 32)             110592
['activation_568[0][0]']


 concatenate_493 (Concatena  (None, 8, 8, 416)            0
['concatenate_492[0][0]',
 te)
'conv2d_621[0][0]']


 batch_normalization_604 (B  (None, 8, 8, 416)            1664
['concatenate_493[0][0]']
 atchNormalization)



 activation_569 (Activation  (None, 8, 8, 416)            0
['batch_normalization_604[0][0
 )                                                                ]'
]


 conv2d_622 (Conv2D)         (None, 8, 8, 32)             119808
['activation_569[0][0]']


 concatenate_494 (Concatena  (None, 8, 8, 448)            0
['concatenate_493[0][0]',
 te)
'conv2d_622[0][0]']


 batch_normalization_605 (B  (None, 8, 8, 448)            1792
```

```
['concatenate_494[0][0]']
 atchNormalization)



 activation_570 (Activation  (None, 8, 8, 448)            0
['batch_normalization_605[0][0
 )                                                                ]'
]


 conv2d_623 (Conv2D)         (None, 8, 8, 32)             129024
['activation_570[0][0]']


 concatenate_495 (Concatena  (None, 8, 8, 480)            0
['concatenate_494[0][0]',
 te)
'conv2d_623[0][0]']


 batch_normalization_606 (B  (None, 8, 8, 480)            1920
['concatenate_495[0][0]']
 atchNormalization)



 activation_571 (Activation  (None, 8, 8, 480)            0
['batch_normalization_606[0][0
 )                                                                ]'
]


 conv2d_624 (Conv2D)         (None, 8, 8, 32)             138240
['activation_571[0][0]']


 concatenate_496 (Concatena  (None, 8, 8, 512)            0
['concatenate_495[0][0]',
 te)
'conv2d_624[0][0]']


 batch_normalization_607 (B  (None, 8, 8, 512)            2048
['concatenate_496[0][0]']
 atchNormalization)



 activation_572 (Activation  (None, 8, 8, 512)            0
```

```
['batch_normalization_607[0][0
 )                                                                  ]'
]


 conv2d_625 (Conv2D)         (None, 8, 8, 32)             147456
['activation_572[0][0]']


 concatenate_497 (Concatena  (None, 8, 8, 544)            0
['concatenate_496[0][0]',
 te)
'conv2d_625[0][0]']


 batch_normalization_608 (B  (None, 8, 8, 544)            2176
['concatenate_497[0][0]']
 atchNormalization)



 activation_573 (Activation  (None, 8, 8, 544)            0
['batch_normalization_608[0][0
 )                                                                  ]'
]


 conv2d_626 (Conv2D)         (None, 8, 8, 32)             156672
['activation_573[0][0]']


 concatenate_498 (Concatena  (None, 8, 8, 576)            0
['concatenate_497[0][0]',
 te)
'conv2d_626[0][0]']


 batch_normalization_609 (B  (None, 8, 8, 576)            2304
['concatenate_498[0][0]']
 atchNormalization)



 activation_574 (Activation  (None, 8, 8, 576)            0
['batch_normalization_609[0][0
 )                                                                  ]'
]


 conv2d_627 (Conv2D)         (None, 8, 8, 32)             165888
```

```
['activation_574[0][0]']


 concatenate_499 (Concatena  (None, 8, 8, 608)            0
['concatenate_498[0][0]',
 te)
'conv2d_627[0][0]']


 batch_normalization_610 (B  (None, 8, 8, 608)            2432
['concatenate_499[0][0]']
 atchNormalization)



 activation_575 (Activation  (None, 8, 8, 608)            0
['batch_normalization_610[0][0
 )                                                                ]'
]


 conv2d_628 (Conv2D)         (None, 8, 8, 32)             175104
['activation_575[0][0]']


 concatenate_500 (Concatena  (None, 8, 8, 640)            0
['concatenate_499[0][0]',
 te)
'conv2d_628[0][0]']


 batch_normalization_611 (B  (None, 8, 8, 640)            2560
['concatenate_500[0][0]']
 atchNormalization)



 activation_576 (Activation  (None, 8, 8, 640)            0
['batch_normalization_611[0][0
 )                                                                ]'
]


 conv2d_629 (Conv2D)         (None, 8, 8, 32)             184320
['activation_576[0][0]']


 concatenate_501 (Concatena  (None, 8, 8, 672)            0
['concatenate_500[0][0]',
 te)
```

```
'conv2d_629[0][0]']


 batch_normalization_612 (B  (None, 8, 8, 672)            2688
['concatenate_501[0][0]']
 atchNormalization)



 activation_577 (Activation  (None, 8, 8, 672)            0
['batch_normalization_612[0][0
 )                                                                ]'
]


 conv2d_630 (Conv2D)         (None, 8, 8, 32)             193536
['activation_577[0][0]']


 concatenate_502 (Concatena  (None, 8, 8, 704)            0
['concatenate_501[0][0]',
 te)
'conv2d_630[0][0]']


 batch_normalization_613 (B  (None, 8, 8, 704)            2816
['concatenate_502[0][0]']
 atchNormalization)



 activation_578 (Activation  (None, 8, 8, 704)            0
['batch_normalization_613[0][0
 )                                                                ]'
]


 conv2d_631 (Conv2D)         (None, 8, 8, 32)             202752
['activation_578[0][0]']


 concatenate_503 (Concatena  (None, 8, 8, 736)            0
['concatenate_502[0][0]',
 te)
'conv2d_631[0][0]']


 batch_normalization_614 (B  (None, 8, 8, 736)            2944
['concatenate_503[0][0]']
 atchNormalization)
```

```
 activation_579 (Activation  (None, 8, 8, 736)            0
['batch_normalization_614[0][0
 )                                                                ]'
]


 conv2d_632 (Conv2D)         (None, 8, 8, 32)             211968
['activation_579[0][0]']


 concatenate_504 (Concatena  (None, 8, 8, 768)            0
['concatenate_503[0][0]',
 te)
'conv2d_632[0][0]']


 batch_normalization_615 (B  (None, 8, 8, 768)            3072
['concatenate_504[0][0]']
 atchNormalization)



 activation_580 (Activation  (None, 8, 8, 768)            0
['batch_normalization_615[0][0
 )                                                                ]'
]


 conv2d_633 (Conv2D)         (None, 8, 8, 32)             221184
['activation_580[0][0]']


 concatenate_505 (Concatena  (None, 8, 8, 800)            0
['concatenate_504[0][0]',
 te)
'conv2d_633[0][0]']


 batch_normalization_616 (B  (None, 8, 8, 800)            3200
['concatenate_505[0][0]']
 atchNormalization)



 activation_581 (Activation  (None, 8, 8, 800)            0
['batch_normalization_616[0][0
 )                                                                ]'
```

```
]


 conv2d_634 (Conv2D)         (None, 8, 8, 32)             230400
['activation_581[0][0]']


 concatenate_506 (Concatena  (None, 8, 8, 832)            0
['concatenate_505[0][0]',
 te)
'conv2d_634[0][0]']


 batch_normalization_617 (B  (None, 8, 8, 832)            3328
['concatenate_506[0][0]']
 atchNormalization)



 activation_582 (Activation  (None, 8, 8, 832)            0
['batch_normalization_617[0][0
 )                                                                ]'
]


 conv2d_635 (Conv2D)         (None, 8, 8, 32)             239616
['activation_582[0][0]']


 concatenate_507 (Concatena  (None, 8, 8, 864)            0
['concatenate_506[0][0]',
 te)
'conv2d_635[0][0]']


 batch_normalization_618 (B  (None, 8, 8, 864)            3456
['concatenate_507[0][0]']
 atchNormalization)



 activation_583 (Activation  (None, 8, 8, 864)            0
['batch_normalization_618[0][0
 )                                                                ]'
]


 conv2d_636 (Conv2D)         (None, 8, 8, 32)             248832
['activation_583[0][0]']
```

```
 concatenate_508 (Concatena  (None, 8, 8, 896)            0
['concatenate_507[0][0]',
 te)
'conv2d_636[0][0]']


 batch_normalization_619 (B  (None, 8, 8, 896)            3584
['concatenate_508[0][0]']
 atchNormalization)



 activation_584 (Activation  (None, 8, 8, 896)            0
['batch_normalization_619[0][0
 )                                                                ]'
]


 conv2d_637 (Conv2D)         (None, 8, 8, 32)             258048
['activation_584[0][0]']


 concatenate_509 (Concatena  (None, 8, 8, 928)            0
['concatenate_508[0][0]',
 te)
'conv2d_637[0][0]']


 batch_normalization_620 (B  (None, 8, 8, 928)            3712
['concatenate_509[0][0]']
 atchNormalization)



 activation_585 (Activation  (None, 8, 8, 928)            0
['batch_normalization_620[0][0
 )                                                                ]'
]


 conv2d_638 (Conv2D)         (None, 8, 8, 32)             267264
['activation_585[0][0]']


 concatenate_510 (Concatena  (None, 8, 8, 960)            0
['concatenate_509[0][0]',
 te)
'conv2d_638[0][0]']
```

```
 batch_normalization_621 (B  (None, 8, 8, 960)            3840
['concatenate_510[0][0]']
 atchNormalization)



 activation_586 (Activation  (None, 8, 8, 960)            0
['batch_normalization_621[0][0
 )                                                                ]'
]


 conv2d_639 (Conv2D)         (None, 8, 8, 32)             276480
['activation_586[0][0]']


 concatenate_511 (Concatena  (None, 8, 8, 992)            0
['concatenate_510[0][0]',
 te)
'conv2d_639[0][0]']


 batch_normalization_622 (B  (None, 8, 8, 992)            3968
['concatenate_511[0][0]']
 atchNormalization)



 activation_587 (Activation  (None, 8, 8, 992)            0
['batch_normalization_622[0][0
 )                                                                ]'
]


 conv2d_640 (Conv2D)         (None, 8, 8, 32)             285696
['activation_587[0][0]']


 concatenate_512 (Concatena  (None, 8, 8, 1024)           0
['concatenate_511[0][0]',
 te)
'conv2d_640[0][0]']


 batch_normalization_623 (B  (None, 8, 8, 1024)           4096
['concatenate_512[0][0]']
 atchNormalization)
```

```
 conv2d_641 (Conv2D)         (None, 8, 8, 512)            524288
['batch_normalization_623[0][0
                                                                 ]'
]


 activation_588 (Activation  (None, 8, 8, 512)            0
['conv2d_641[0][0]']
 )



 average_pooling2d_26 (Aver  (None, 4, 4, 512)            0
['activation_588[0][0]']
 agePooling2D)



 batch_normalization_624 (B  (None, 4, 4, 512)            2048
['average_pooling2d_26[0][0]']
 atchNormalization)



 activation_589 (Activation  (None, 4, 4, 512)            0
['batch_normalization_624[0][0
 )                                                               ]'
]


 conv2d_642 (Conv2D)         (None, 4, 4, 32)             147456
['activation_589[0][0]']


 concatenate_513 (Concatena  (None, 4, 4, 544)            0
['average_pooling2d_26[0][0]',
 te)
'conv2d_642[0][0]']


 batch_normalization_625 (B  (None, 4, 4, 544)            2176
['concatenate_513[0][0]']
 atchNormalization)



 activation_590 (Activation  (None, 4, 4, 544)            0
['batch_normalization_625[0][0
 )                                                               ]'
]
```

```
 conv2d_643 (Conv2D)         (None, 4, 4, 32)             156672
['activation_590[0][0]']


 concatenate_514 (Concatena  (None, 4, 4, 576)            0
['concatenate_513[0][0]',
 te)
'conv2d_643[0][0]']


 batch_normalization_626 (B  (None, 4, 4, 576)            2304
['concatenate_514[0][0]']
 atchNormalization)



 activation_591 (Activation  (None, 4, 4, 576)            0
['batch_normalization_626[0][0
 )                                                                ]'
]


 conv2d_644 (Conv2D)         (None, 4, 4, 32)             165888
['activation_591[0][0]']


 concatenate_515 (Concatena  (None, 4, 4, 608)            0
['concatenate_514[0][0]',
 te)
'conv2d_644[0][0]']


 batch_normalization_627 (B  (None, 4, 4, 608)            2432
['concatenate_515[0][0]']
 atchNormalization)



 activation_592 (Activation  (None, 4, 4, 608)            0
['batch_normalization_627[0][0
 )                                                                ]'
]


 conv2d_645 (Conv2D)         (None, 4, 4, 32)             175104
['activation_592[0][0]']
```

```
 concatenate_516 (Concatena  (None, 4, 4, 640)            0
['concatenate_515[0][0]',
 te)
'conv2d_645[0][0]']


 batch_normalization_628 (B  (None, 4, 4, 640)            2560
['concatenate_516[0][0]']
 atchNormalization)



 activation_593 (Activation  (None, 4, 4, 640)            0
['batch_normalization_628[0][0
 )                                                                ]'
]


 conv2d_646 (Conv2D)         (None, 4, 4, 32)             184320
['activation_593[0][0]']


 concatenate_517 (Concatena  (None, 4, 4, 672)            0
['concatenate_516[0][0]',
 te)
'conv2d_646[0][0]']


 batch_normalization_629 (B  (None, 4, 4, 672)            2688
['concatenate_517[0][0]']
 atchNormalization)



 activation_594 (Activation  (None, 4, 4, 672)            0
['batch_normalization_629[0][0
 )                                                                ]'
]


 conv2d_647 (Conv2D)         (None, 4, 4, 32)             193536
['activation_594[0][0]']


 concatenate_518 (Concatena  (None, 4, 4, 704)            0
['concatenate_517[0][0]',
 te)
'conv2d_647[0][0]']
```

```
 batch_normalization_630 (B  (None, 4, 4, 704)            2816
['concatenate_518[0][0]']
 atchNormalization)



 activation_595 (Activation  (None, 4, 4, 704)            0
['batch_normalization_630[0][0
 )                                                                ]'
]


 conv2d_648 (Conv2D)         (None, 4, 4, 32)             202752
['activation_595[0][0]']


 concatenate_519 (Concatena  (None, 4, 4, 736)            0
['concatenate_518[0][0]',
 te)
'conv2d_648[0][0]']


 batch_normalization_631 (B  (None, 4, 4, 736)            2944
['concatenate_519[0][0]']
 atchNormalization)



 activation_596 (Activation  (None, 4, 4, 736)            0
['batch_normalization_631[0][0
 )                                                                ]'
]


 conv2d_649 (Conv2D)         (None, 4, 4, 32)             211968
['activation_596[0][0]']


 concatenate_520 (Concatena  (None, 4, 4, 768)            0
['concatenate_519[0][0]',
 te)
'conv2d_649[0][0]']


 batch_normalization_632 (B  (None, 4, 4, 768)            3072
['concatenate_520[0][0]']
 atchNormalization)
```

```
 activation_597 (Activation  (None, 4, 4, 768)            0
['batch_normalization_632[0][0
 )                                                                ]'
]


 conv2d_650 (Conv2D)         (None, 4, 4, 32)             221184
['activation_597[0][0]']


 concatenate_521 (Concatena  (None, 4, 4, 800)            0
['concatenate_520[0][0]',
 te)
'conv2d_650[0][0]']


 batch_normalization_633 (B  (None, 4, 4, 800)            3200
['concatenate_521[0][0]']
 atchNormalization)



 activation_598 (Activation  (None, 4, 4, 800)            0
['batch_normalization_633[0][0
 )                                                                ]'
]


 conv2d_651 (Conv2D)         (None, 4, 4, 32)             230400
['activation_598[0][0]']


 concatenate_522 (Concatena  (None, 4, 4, 832)            0
['concatenate_521[0][0]',
 te)
'conv2d_651[0][0]']


 batch_normalization_634 (B  (None, 4, 4, 832)            3328
['concatenate_522[0][0]']
 atchNormalization)



 activation_599 (Activation  (None, 4, 4, 832)            0
['batch_normalization_634[0][0
 )                                                                ]'
]
```

```
 conv2d_652 (Conv2D)         (None, 4, 4, 32)             239616
['activation_599[0][0]']


 concatenate_523 (Concatena  (None, 4, 4, 864)            0
['concatenate_522[0][0]',
 te)
'conv2d_652[0][0]']


 batch_normalization_635 (B  (None, 4, 4, 864)            3456
['concatenate_523[0][0]']
 atchNormalization)



 activation_600 (Activation  (None, 4, 4, 864)            0
['batch_normalization_635[0][0
 )                                                                ]'
]


 conv2d_653 (Conv2D)         (None, 4, 4, 32)             248832
['activation_600[0][0]']


 concatenate_524 (Concatena  (None, 4, 4, 896)            0
['concatenate_523[0][0]',
 te)
'conv2d_653[0][0]']


 batch_normalization_636 (B  (None, 4, 4, 896)            3584
['concatenate_524[0][0]']
 atchNormalization)



 activation_601 (Activation  (None, 4, 4, 896)            0
['batch_normalization_636[0][0
 )                                                                ]'
]


 conv2d_654 (Conv2D)         (None, 4, 4, 32)             258048
['activation_601[0][0]']


 concatenate_525 (Concatena  (None, 4, 4, 928)            0
['concatenate_524[0][0]',
```

```
 te)                                                                                                  'conv2d_654[0][0]']


 batch_normalization_637 (B  (None, 4, 4, 928)            3712        ['concatenate_525[0][0]']
 atchNormalization)


 activation_602 (Activation  (None, 4, 4, 928)            0           ['batch_normalization_637[0][0
 )                                                                                                    ]']


 conv2d_655 (Conv2D)         (None, 4, 4, 32)             267264      ['activation_602[0][0]']


 concatenate_526 (Concatena  (None, 4, 4, 960)            0           ['concatenate_525[0][0]',
 te)                                                                  'conv2d_655[0][0]']


 batch_normalization_638 (B  (None, 4, 4, 960)            3840        ['concatenate_526[0][0]']
 atchNormalization)


 activation_603 (Activation  (None, 4, 4, 960)            0           ['batch_normalization_638[0][0
 )                                                                                                    ]']


 conv2d_656 (Conv2D)         (None, 4, 4, 32)             276480      ['activation_603[0][0]']


 concatenate_527 (Concatena  (None, 4, 4, 992)            0           ['concatenate_526[0][0]',
 te)                                                                  'conv2d_656[0][0]']


 batch_normalization_639 (B  (None, 4, 4, 992)            3968        ['concatenate_527[0][0]']
```

```
 atchNormalization)


 activation_604 (Activation  (None, 4, 4, 992)            0
['batch_normalization_639[0][0
 )                                                                ]'
]


 conv2d_657 (Conv2D)         (None, 4, 4, 32)             285696
['activation_604[0][0]']


 concatenate_528 (Concatena  (None, 4, 4, 1024)           0
['concatenate_527[0][0]',
 te)
'conv2d_657[0][0]']


 batch_normalization_640 (B  (None, 4, 4, 1024)           4096
['concatenate_528[0][0]']
 atchNormalization)


 activation_605 (Activation  (None, 4, 4, 1024)           0
['batch_normalization_640[0][0
 )                                                                ]'
]


 conv2d_658 (Conv2D)         (None, 4, 4, 1)              1025
['activation_605[0][0]']


 up_sampling2d_25 (UpSampli  (None, 128, 128, 1)          0
['conv2d_658[0][0]']
 ng2D)


==================================================================
============================
Total params: 9913921 (37.82 MB)
Trainable params: 9845121 (37.56 MB)
Non-trainable params: 68800 (268.75 KB)
__________________________________________________________________
_____________________________
```

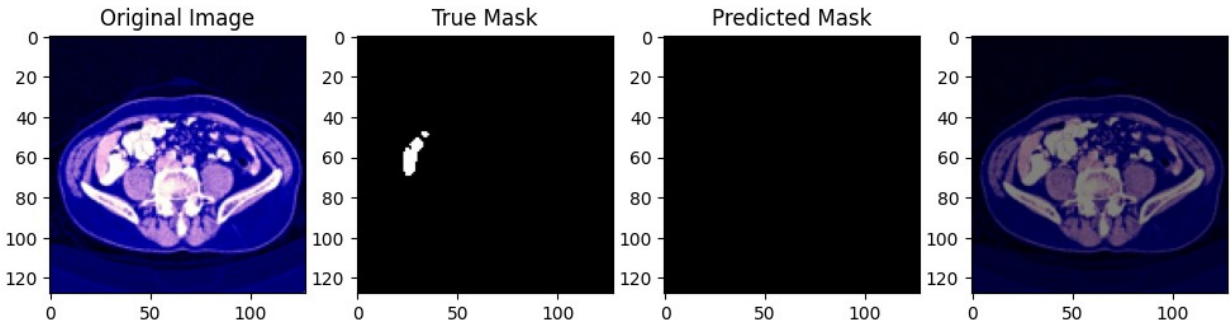
```
model5.compile(optimizer=Adam(lr=0.001, beta_1=0.9,
beta_2=0.999),loss= 'binary_crossentropy', metrics=[dice_coefficient])
model5.save('Densenet_model5.h5')

checkpoint = tf.keras.callbacks.ModelCheckpoint("Densenet_model5.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                   patience=5,
                                                   mode='auto',

restore_best_weights=True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                  factor=0.2,
                                                  patience=2,
                                                  min_lr=1e-6,
                                                  mode='auto',
                                                  verbose=1)
# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]

history5= model5.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model5.save('Densenet_model5.h5')

Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.0623 -
dice_coefficient: 0.9790
Epoch 1: val_loss improved from inf to 0.12298, saving model to
Densenet_model5.h5
394/394 [==============================] - 57s 59ms/step - loss:
0.0623 - dice_coefficient: 0.9790 - val_loss: 0.1230 -
val_dice_coefficient: 0.9451 - lr: 0.0010
Epoch 2/10
394/394 [==============================] - ETA: 0s - loss: 0.0448 -
dice_coefficient: 0.9836
Epoch 2: val_loss improved from 0.12298 to 0.04596, saving model to
Densenet_model5.h5
394/394 [==============================] - 21s 54ms/step - loss:
0.0448 - dice_coefficient: 0.9836 - val_loss: 0.0460 -
val_dice_coefficient: 0.9877 - lr: 0.0010
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: 0.0428 -
dice_coefficient: 0.9841
Epoch 3: val_loss improved from 0.04596 to 0.04147, saving model to
Densenet_model5.h5
394/394 [==============================] - 21s 53ms/step - loss:
```

```
0.0428 - dice_coefficient: 0.9841 - val_loss: 0.0415 -
val_dice_coefficient: 0.9832 - lr: 0.0010
Epoch 4/10
393/394 [============================>.] - ETA: 0s - loss: 0.0414 -
dice_coefficient: 0.9844
Epoch 4: val_loss improved from 0.04147 to 0.03852, saving model to
Densenet_model5.h5
394/394 [==============================] - 21s 53ms/step - loss:
0.0414 - dice_coefficient: 0.9844 - val_loss: 0.0385 -
val_dice_coefficient: 0.9860 - lr: 0.0010
Epoch 5/10
394/394 [==============================] - ETA: 0s - loss: 0.0407 -
dice_coefficient: 0.9844
Epoch 5: val_loss did not improve from 0.03852
394/394 [==============================] - 20s 52ms/step - loss:
0.0407 - dice_coefficient: 0.9844 - val_loss: 0.0463 -
val_dice_coefficient: 0.9872 - lr: 0.0010
Epoch 6/10
393/394 [============================>.] - ETA: 0s - loss: 0.0398 -
dice_coefficient: 0.9845
Epoch 6: val_loss improved from 0.03852 to 0.03789, saving model to
Densenet_model5.h5
394/394 [==============================] - 21s 54ms/step - loss:
0.0397 - dice_coefficient: 0.9845 - val_loss: 0.0379 -
val_dice_coefficient: 0.9854 - lr: 0.0010
Epoch 7/10
393/394 [============================>.] - ETA: 0s - loss: 0.0393 -
dice_coefficient: 0.9845
Epoch 7: val_loss did not improve from 0.03789
394/394 [==============================] - 20s 52ms/step - loss:
0.0393 - dice_coefficient: 0.9845 - val_loss: 0.0409 -
val_dice_coefficient: 0.9819 - lr: 0.0010
Epoch 8/10
393/394 [============================>.] - ETA: 0s - loss: 0.0385 -
dice_coefficient: 0.9847
Epoch 8: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.

Epoch 8: val_loss did not improve from 0.03789
394/394 [==============================] - 22s 55ms/step - loss:
0.0385 - dice_coefficient: 0.9847 - val_loss: 0.0406 -
val_dice_coefficient: 0.9847 - lr: 0.0010
Epoch 9/10
394/394 [==============================] - ETA: 0s - loss: 0.0374 -
dice_coefficient: 0.9848
Epoch 9: val_loss improved from 0.03789 to 0.03591, saving model to
Densenet_model5.h5
394/394 [==============================] - 22s 56ms/step - loss:
0.0374 - dice_coefficient: 0.9848 - val_loss: 0.0359 -
```

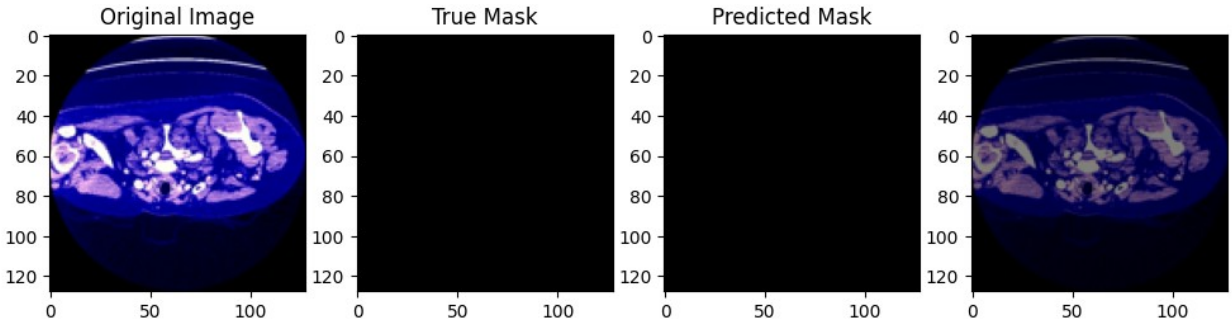
```
val_dice_coefficient: 0.9860 - lr: 2.0000e-04
Epoch 10/10
393/394 [============================>.] - ETA: 0s - loss: 0.0370 -
dice_coefficient: 0.9849
Epoch 10: val_loss improved from 0.03591 to 0.03520, saving model to
Densenet_model5.h5
394/394 [==============================] - 22s 57ms/step - loss:
0.0371 - dice_coefficient: 0.9849 - val_loss: 0.0352 -
val_dice_coefficient: 0.9858 - lr: 2.0000e-04

scores5 = model5.evaluate(x_valid, y_valid)
scores5[1]

57/57 [==============================] - 3s 23ms/step - loss: 0.0360 -
dice_coefficient: 0.9858

0.9857683181762695

prediction5 = model5.predict(x_test)
test_scores5 = model5.evaluate(x_test, y_test)
test_scores5[1]

29/29 [==============================] - 2s 20ms/step
29/29 [==============================] - 1s 23ms/step - loss: 0.0385 -
dice_coefficient: 0.9846

0.9846317768096924

import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 20, replace=False)
for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model3
    predicted_mask = model5.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
    threshold = 0.5  # Adjust this threshold based on your model's
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
```

```
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')
    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
'nipy_spectral')
    plt.show()
```

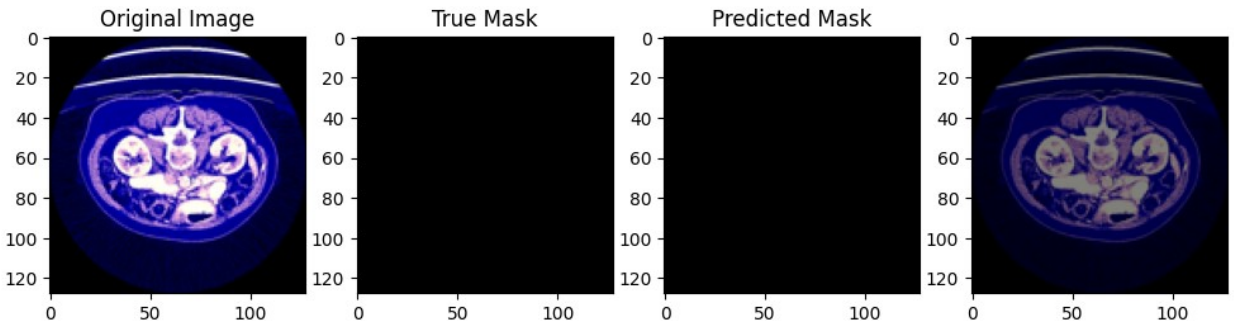
```
333
1/1 [==============================] - 0s 28ms/step
```



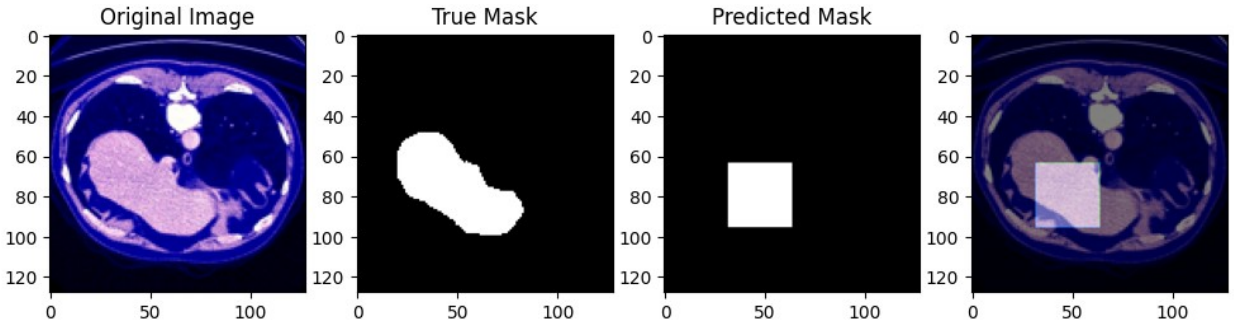

```
812
1/1 [==============================] - 0s 28ms/step
```



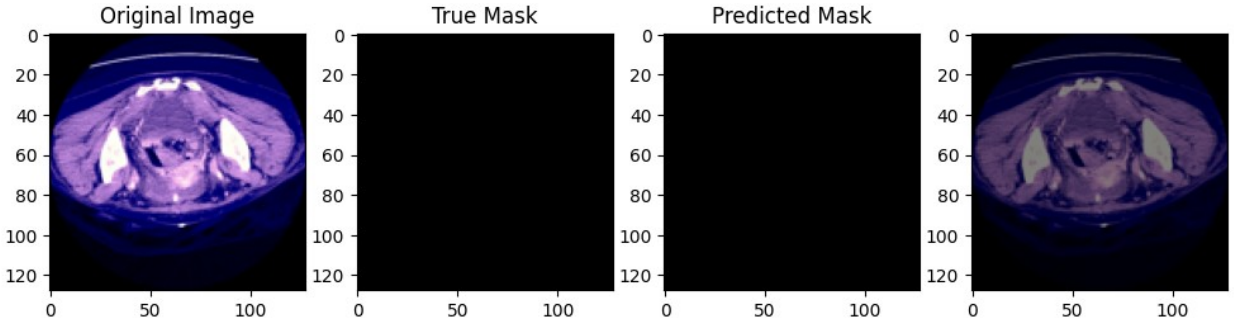
```
745
1/1 [==============================] - 0s 26ms/step
```



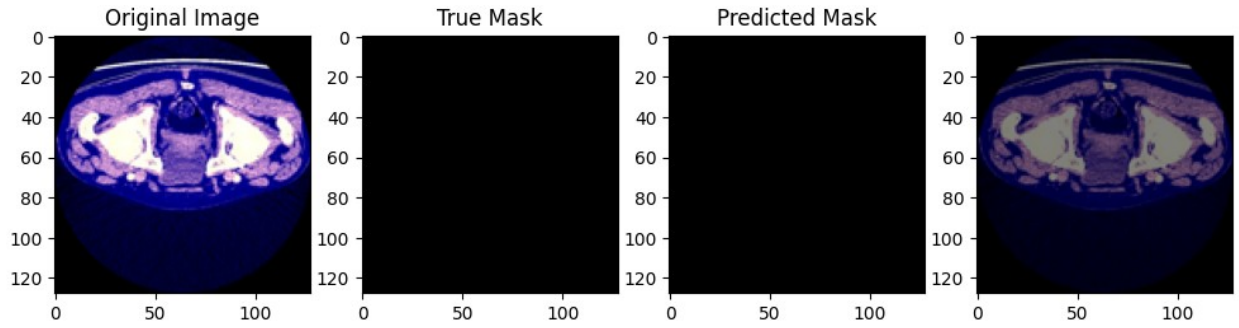

```
245
1/1 [==============================] - 0s 28ms/step
```



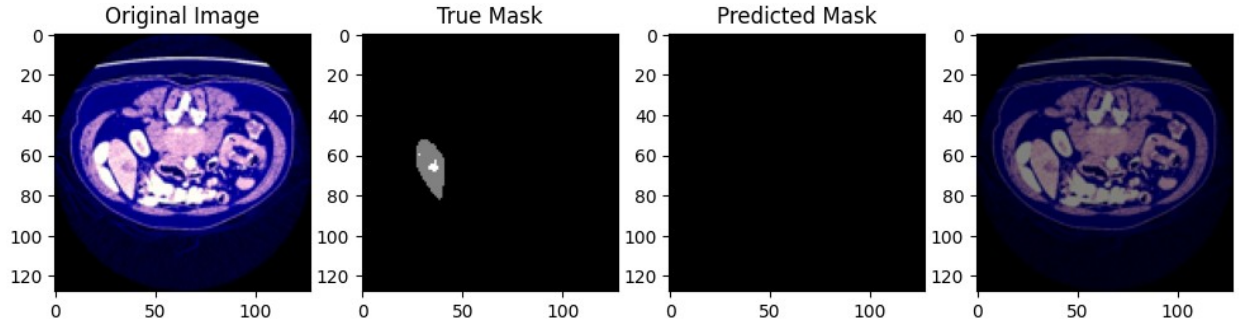

```
40
1/1 [==============================] - 0s 26ms/step
```



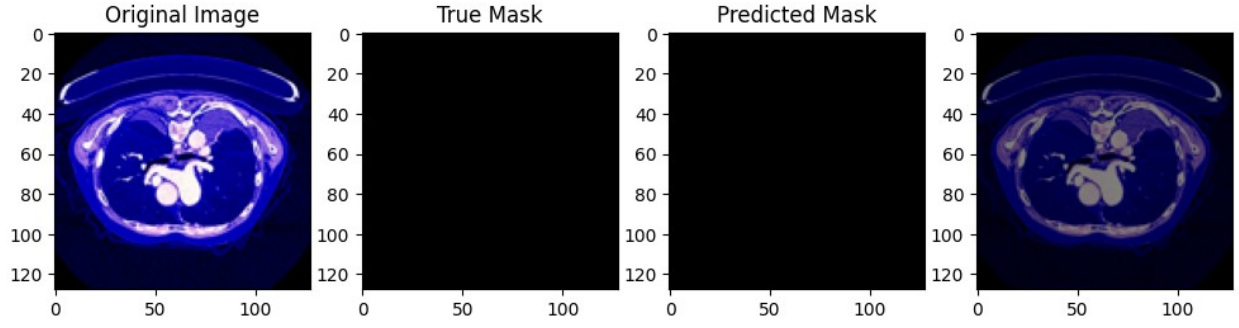

```
427
1/1 [==============================] - 0s 27ms/step
```



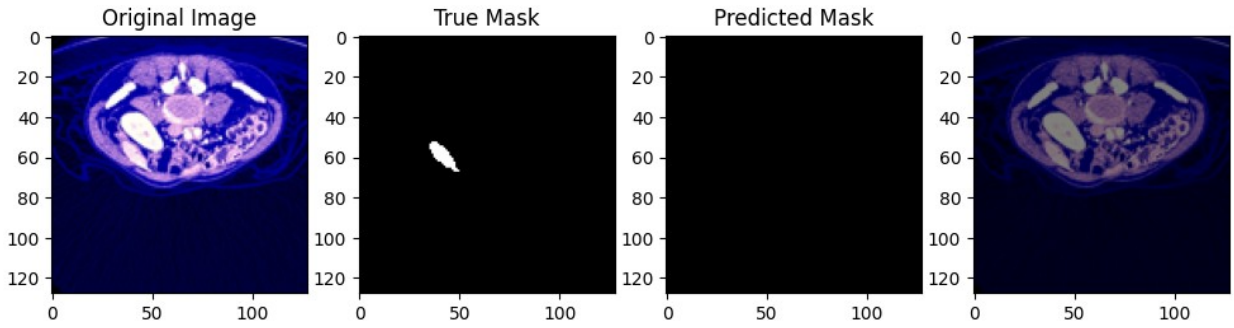

```
303
1/1 [==============================] - 0s 26ms/step
```



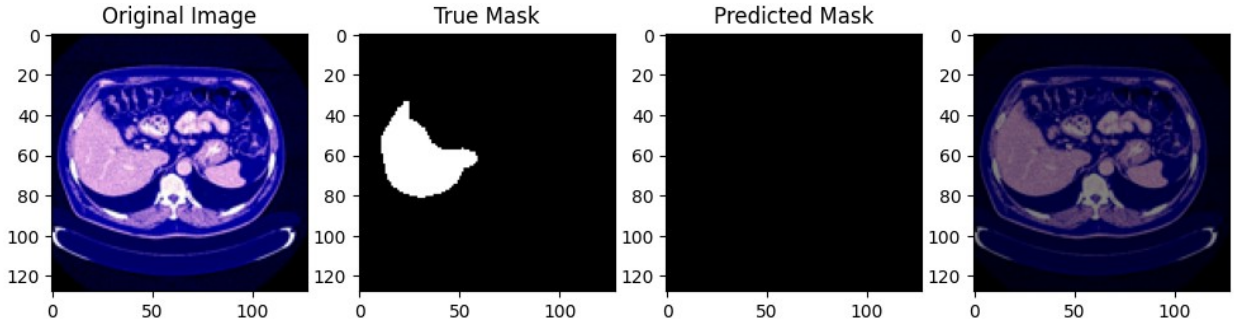

```
588
1/1 [==============================] - 0s 26ms/step
```



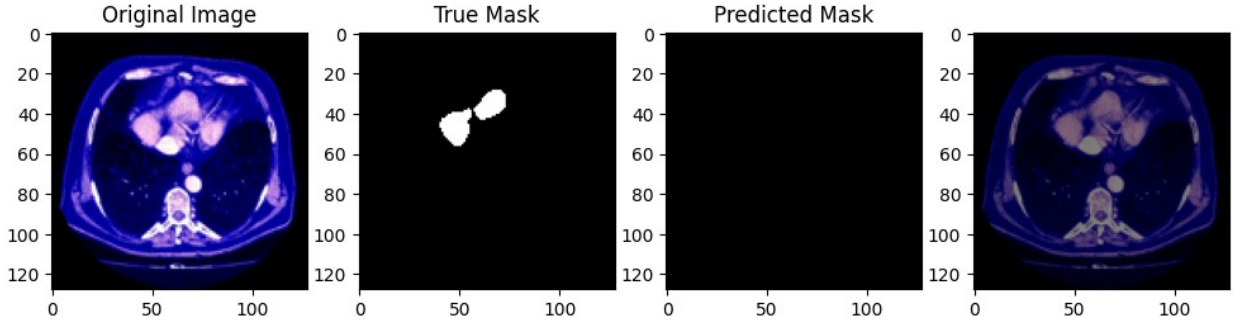

```
210
1/1 [==============================] - 0s 37ms/step
```



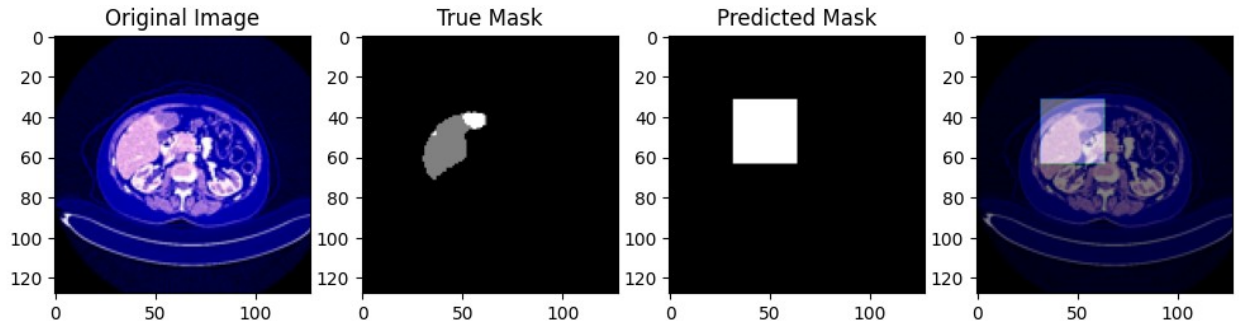

```
137
1/1 [==============================] - 0s 30ms/step
```



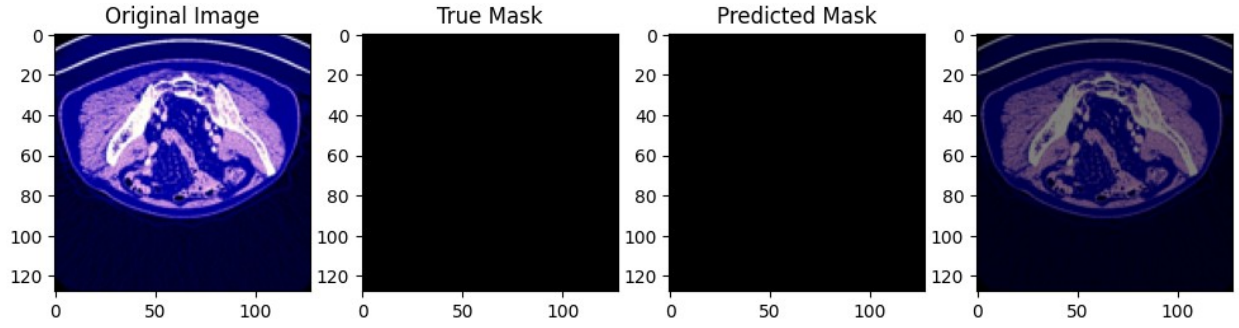

```
138
1/1 [==============================] - 0s 27ms/step
```



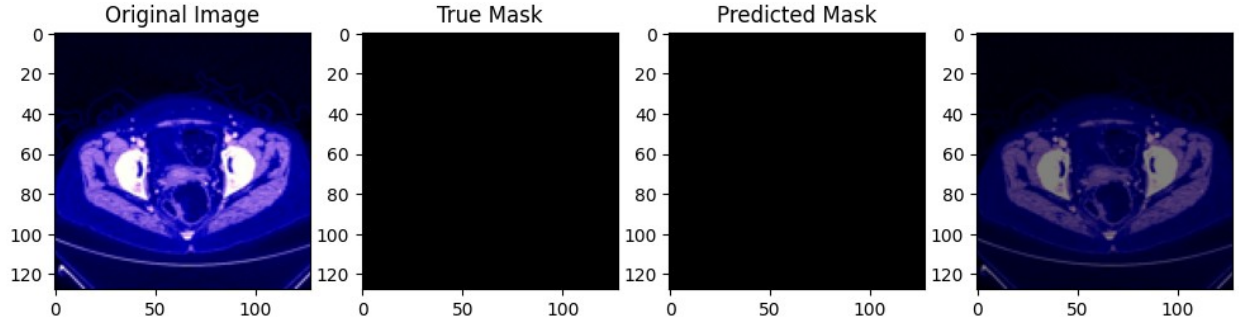

```
261
1/1 [==============================] - 0s 26ms/step
```



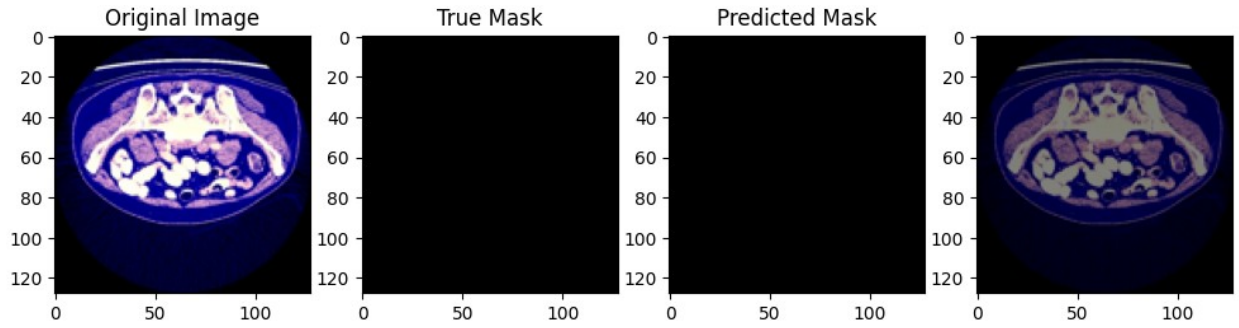

```
352
1/1 [==============================] - 0s 26ms/step
```



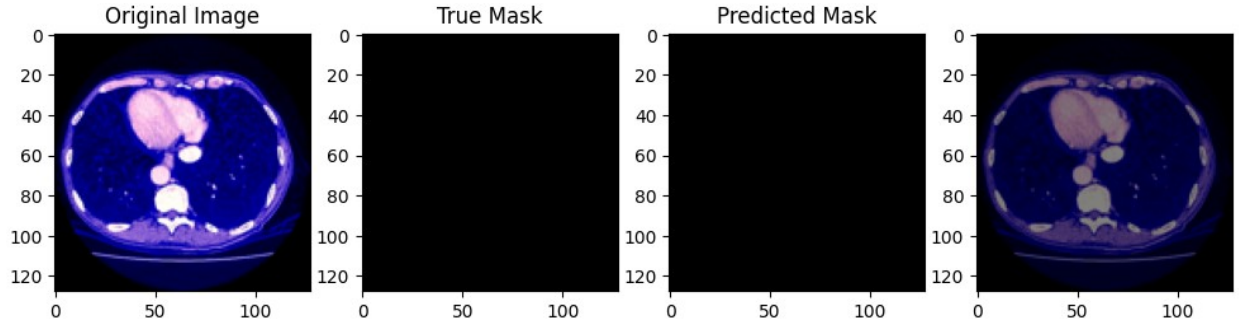

```
360
1/1 [==============================] - 0s 27ms/step
```



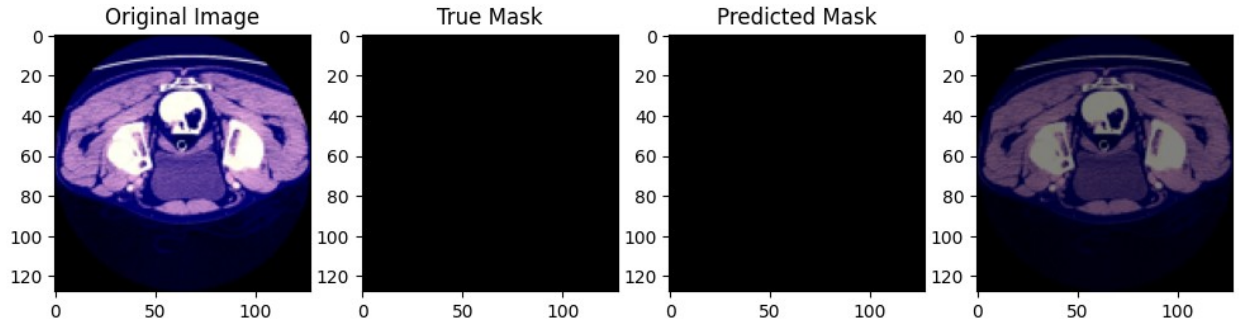

```
347
1/1 [==============================] - 0s 31ms/step
```



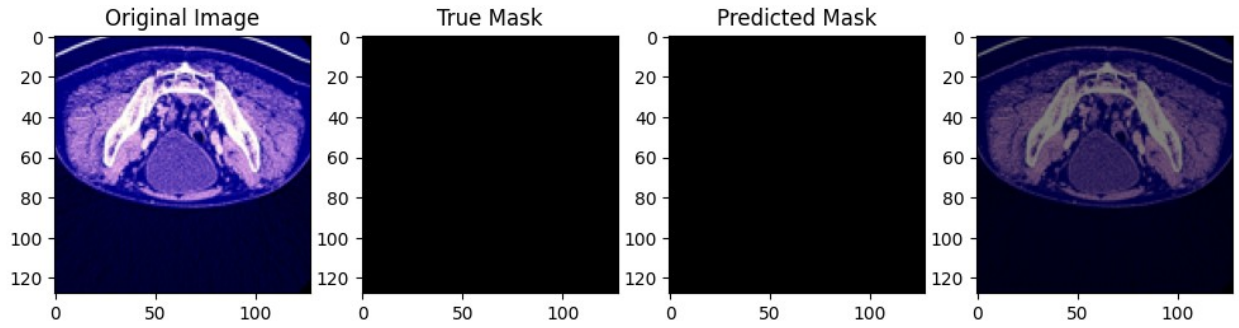

```
479
1/1 [==============================] - 0s 31ms/step
```



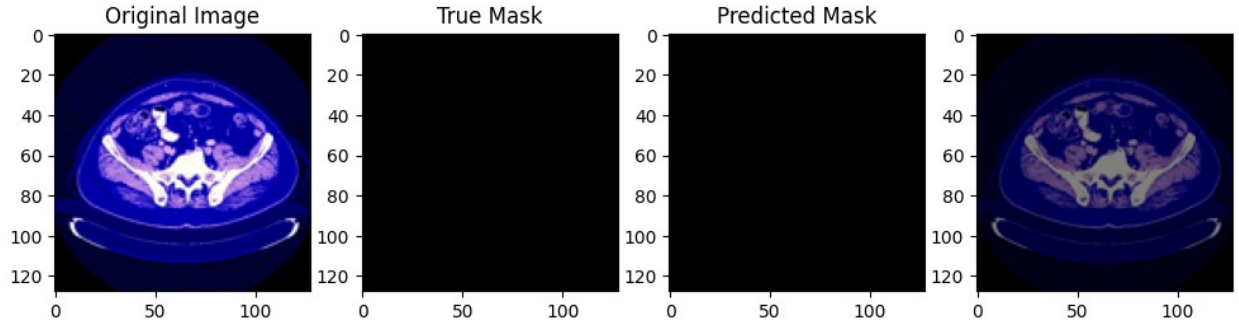

```
307
1/1 [==============================] - 0s 33ms/step
```



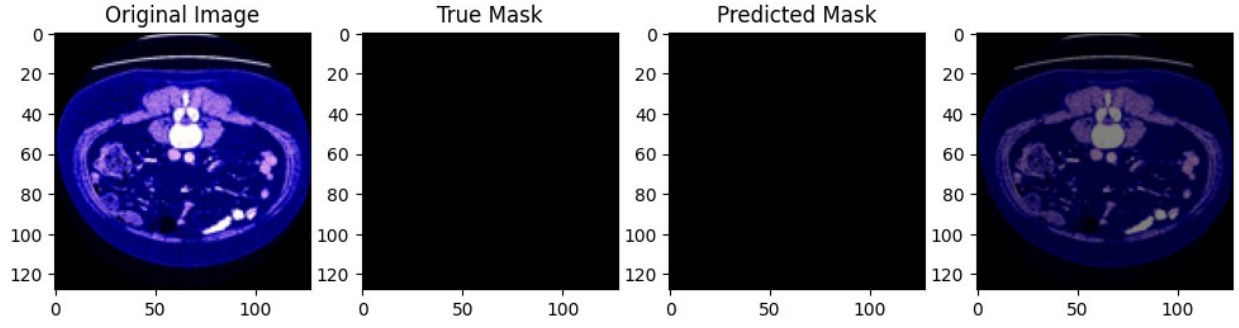

```
658
1/1 [==============================] - 0s 30ms/step
```



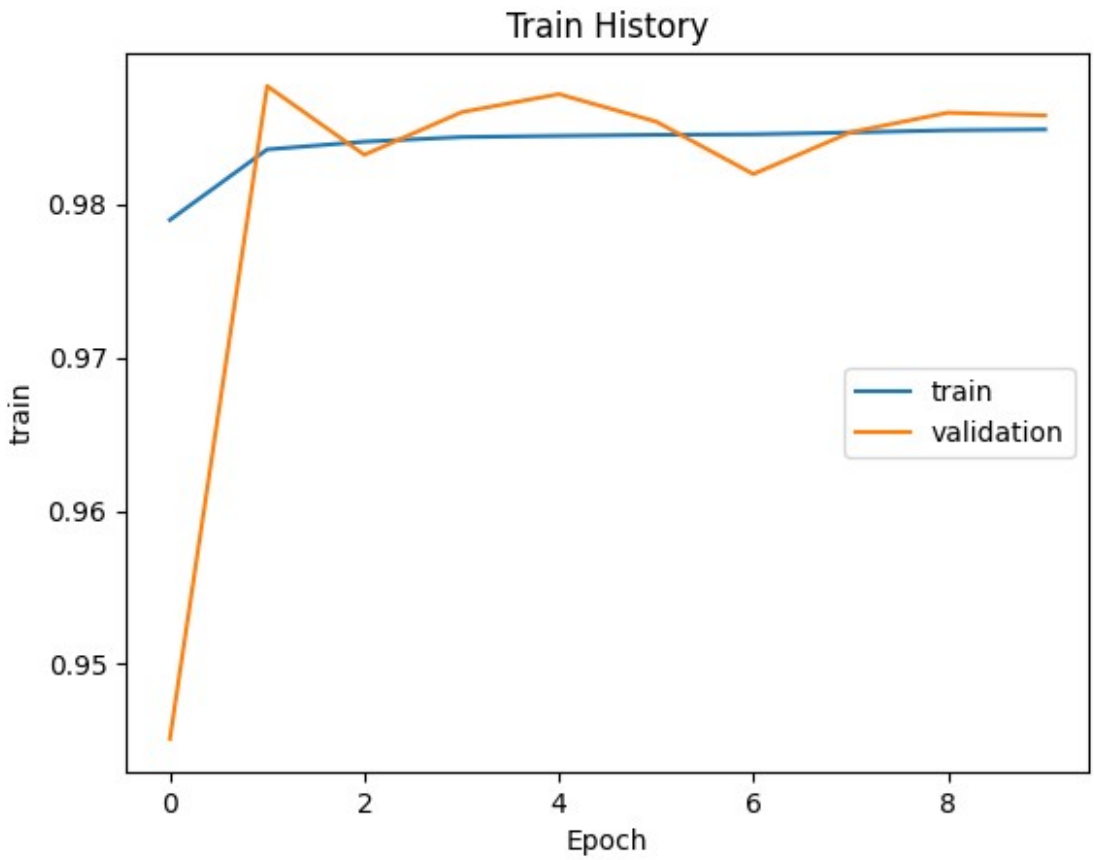

```
895
1/1 [==============================] - 0s 30ms/step
```



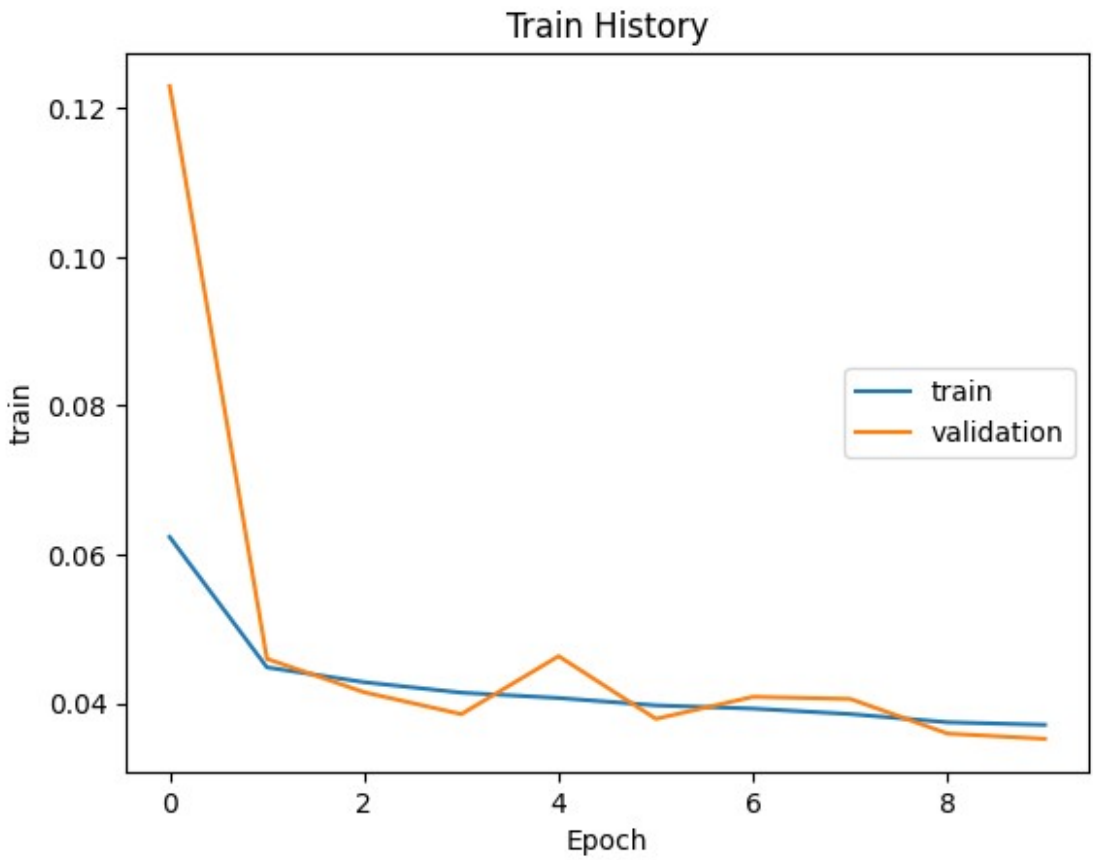

```
111
1/1 [==============================] - 0s 29ms/step
```




```
show_history(history5, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history5, 'loss', 'val_loss')
```




```
import tensorflow as tf
from tensorflow import keras

def conv_block(inputs, filters, kernel_size=(3, 3), padding='same',
activation='relu'):
    conv = keras.layers.Conv2D(filters, kernel_size, padding=padding)
(inputs)
    conv = keras.layers.BatchNormalization()(conv)
    conv = keras.layers.Activation(activation)(conv)
    conv = keras.layers.Conv2D(filters, kernel_size, padding=padding)
(conv)
    conv = keras.layers.BatchNormalization()(conv)
    conv = keras.layers.Activation(activation)(conv)
    return conv

def down_sampling_block(inputs, filters):
    down = keras.layers.MaxPooling2D((2, 2))(inputs)
    down = conv_block(down, filters)
    return down

def up_sampling_block(inputs, skip_connections, filters):
    up = keras.layers.UpSampling2D((2, 2))(inputs)
    concat = keras.layers.Concatenate()([up, skip_connections])
```

```
    up_conv = conv_block(concat, filters)
    return up_conv

def unetpp(input_shape=(128, 128, 3)):
    inputs = keras.layers.Input(input_shape)

    # Encoder
    conv1 = conv_block(inputs, 64)
    pool1 = down_sampling_block(conv1, 128)
    conv2 = conv_block(pool1, 128)
    pool2 = down_sampling_block(conv2, 256)
    conv3 = conv_block(pool2, 256)
    pool3 = down_sampling_block(conv3, 512)
    conv4 = conv_block(pool3, 512)

    # Decoder
    up1 = up_sampling_block(conv4, conv3, 256)
    up2 = up_sampling_block(up1, conv2, 128)
    up3 = up_sampling_block(up2, conv1, 64)

    outputs = keras.layers.Conv2D(1, (1, 1), activation='sigmoid')
(up3)

    model = keras.models.Model(inputs, outputs)
    return model

# Instantiate the model
model6 = unetpp()

# Compile the model
model6.compile(optimizer='Adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Display the model summary
model6.summary()
```

```
Model: "model_13"
__________________________________________________________________________________
_________________________________
 Layer (type)                Output Shape              Param #
Connected to
==================================================================================
=================================
 input_14 (InputLayer)       [(None, 128, 128, 3)]     0         []



 conv2d_680 (Conv2D)         (None, 128, 128, 64)      1792
['input_14[0][0]']
```

```
 batch_normalization_661 (B  (None, 128, 128, 64)         256
['conv2d_680[0][0]']
 atchNormalization)



 activation_626 (Activation  (None, 128, 128, 64)         0
['batch_normalization_661[0][0
 )                                                                ]'
]


 conv2d_681 (Conv2D)         (None, 128, 128, 64)         36928
['activation_626[0][0]']


 batch_normalization_662 (B  (None, 128, 128, 64)         256
['conv2d_681[0][0]']
 atchNormalization)



 activation_627 (Activation  (None, 128, 128, 64)         0
['batch_normalization_662[0][0
 )                                                                ]'
]


 max_pooling2d_20 (MaxPooli  (None, 64, 64, 64)           0
['activation_627[0][0]']
 ng2D)



 conv2d_682 (Conv2D)         (None, 64, 64, 128)          73856
['max_pooling2d_20[0][0]']


 batch_normalization_663 (B  (None, 64, 64, 128)          512
['conv2d_682[0][0]']
 atchNormalization)



 activation_628 (Activation  (None, 64, 64, 128)          0
['batch_normalization_663[0][0
 )                                                                ]'
]
```

```
 conv2d_683 (Conv2D)         (None, 64, 64, 128)          147584
['activation_628[0][0]']


 batch_normalization_664 (B  (None, 64, 64, 128)          512
['conv2d_683[0][0]']
 atchNormalization)



 activation_629 (Activation  (None, 64, 64, 128)          0
['batch_normalization_664[0][0
 )                                                                ]'
]


 conv2d_684 (Conv2D)         (None, 64, 64, 128)          147584
['activation_629[0][0]']


 batch_normalization_665 (B  (None, 64, 64, 128)          512
['conv2d_684[0][0]']
 atchNormalization)



 activation_630 (Activation  (None, 64, 64, 128)          0
['batch_normalization_665[0][0
 )                                                                ]'
]


 conv2d_685 (Conv2D)         (None, 64, 64, 128)          147584
['activation_630[0][0]']


 batch_normalization_666 (B  (None, 64, 64, 128)          512
['conv2d_685[0][0]']
 atchNormalization)



 activation_631 (Activation  (None, 64, 64, 128)          0
['batch_normalization_666[0][0
 )                                                                ]'
]


 max_pooling2d_21 (MaxPooli  (None, 32, 32, 128)          0
['activation_631[0][0]']
```

```
 ng2D)


 conv2d_686 (Conv2D)         (None, 32, 32, 256)          295168
['max_pooling2d_21[0][0]']


 batch_normalization_667 (B  (None, 32, 32, 256)          1024
['conv2d_686[0][0]']
 atchNormalization)


 activation_632 (Activation  (None, 32, 32, 256)          0
['batch_normalization_667[0][0
 )                                                                ]'
]


 conv2d_687 (Conv2D)         (None, 32, 32, 256)          590080
['activation_632[0][0]']


 batch_normalization_668 (B  (None, 32, 32, 256)          1024
['conv2d_687[0][0]']
 atchNormalization)


 activation_633 (Activation  (None, 32, 32, 256)          0
['batch_normalization_668[0][0
 )                                                                ]'
]


 conv2d_688 (Conv2D)         (None, 32, 32, 256)          590080
['activation_633[0][0]']


 batch_normalization_669 (B  (None, 32, 32, 256)          1024
['conv2d_688[0][0]']
 atchNormalization)


 activation_634 (Activation  (None, 32, 32, 256)          0
['batch_normalization_669[0][0
 )                                                                ]'
]
```

```
 conv2d_689 (Conv2D)         (None, 32, 32, 256)          590080
['activation_634[0][0]']


 batch_normalization_670 (B  (None, 32, 32, 256)          1024
['conv2d_689[0][0]']
 atchNormalization)



 activation_635 (Activation  (None, 32, 32, 256)          0
['batch_normalization_670[0][0
 )                                                                ]'
]


 max_pooling2d_22 (MaxPooli  (None, 16, 16, 256)          0
['activation_635[0][0]']
 ng2D)



 conv2d_690 (Conv2D)         (None, 16, 16, 512)          1180160
['max_pooling2d_22[0][0]']


 batch_normalization_671 (B  (None, 16, 16, 512)          2048
['conv2d_690[0][0]']
 atchNormalization)



 activation_636 (Activation  (None, 16, 16, 512)          0
['batch_normalization_671[0][0
 )                                                                ]'
]


 conv2d_691 (Conv2D)         (None, 16, 16, 512)          2359808
['activation_636[0][0]']


 batch_normalization_672 (B  (None, 16, 16, 512)          2048
['conv2d_691[0][0]']
 atchNormalization)



 activation_637 (Activation  (None, 16, 16, 512)          0
```

```
['batch_normalization_672[0][0
 )                                                                ]'
]


 conv2d_692 (Conv2D)         (None, 16, 16, 512)          2359808
['activation_637[0][0]']


 batch_normalization_673 (B  (None, 16, 16, 512)          2048
['conv2d_692[0][0]']
 atchNormalization)



 activation_638 (Activation  (None, 16, 16, 512)          0
['batch_normalization_673[0][0
 )                                                                ]'
]


 conv2d_693 (Conv2D)         (None, 16, 16, 512)          2359808
['activation_638[0][0]']


 batch_normalization_674 (B  (None, 16, 16, 512)          2048
['conv2d_693[0][0]']
 atchNormalization)



 activation_639 (Activation  (None, 16, 16, 512)          0
['batch_normalization_674[0][0
 )                                                                ]'
]


 up_sampling2d_29 (UpSampli  (None, 32, 32, 512)          0
['activation_639[0][0]']
 ng2D)



 concatenate_532 (Concatena  (None, 32, 32, 768)          0
['up_sampling2d_29[0][0]',
 te)
'activation_635[0][0]']


 conv2d_694 (Conv2D)         (None, 32, 32, 256)          1769728
```

```
['concatenate_532[0][0]']


 batch_normalization_675 (B  (None, 32, 32, 256)          1024
['conv2d_694[0][0]']
 atchNormalization)



 activation_640 (Activation  (None, 32, 32, 256)          0
['batch_normalization_675[0][0
 )                                                                ]'
]


 conv2d_695 (Conv2D)         (None, 32, 32, 256)          590080
['activation_640[0][0]']


 batch_normalization_676 (B  (None, 32, 32, 256)          1024
['conv2d_695[0][0]']
 atchNormalization)



 activation_641 (Activation  (None, 32, 32, 256)          0
['batch_normalization_676[0][0
 )                                                                ]'
]


 up_sampling2d_30 (UpSampli  (None, 64, 64, 256)          0
['activation_641[0][0]']
 ng2D)



 concatenate_533 (Concatena  (None, 64, 64, 384)          0
['up_sampling2d_30[0][0]',
 te)
'activation_631[0][0]']


 conv2d_696 (Conv2D)         (None, 64, 64, 128)          442496
['concatenate_533[0][0]']


 batch_normalization_677 (B  (None, 64, 64, 128)          512
['conv2d_696[0][0]']
 atchNormalization)
```

```
 activation_642 (Activation  (None, 64, 64, 128)          0
['batch_normalization_677[0][0
 )                                                                ]'
]


 conv2d_697 (Conv2D)         (None, 64, 64, 128)          147584
['activation_642[0][0]']


 batch_normalization_678 (B  (None, 64, 64, 128)          512
['conv2d_697[0][0]']
 atchNormalization)



 activation_643 (Activation  (None, 64, 64, 128)          0
['batch_normalization_678[0][0
 )                                                                ]'
]


 up_sampling2d_31 (UpSampli  (None, 128, 128, 128)        0
['activation_643[0][0]']
 ng2D)



 concatenate_534 (Concatena  (None, 128, 128, 192)        0
['up_sampling2d_31[0][0]',
 te)
'activation_627[0][0]']


 conv2d_698 (Conv2D)         (None, 128, 128, 64)         110656
['concatenate_534[0][0]']


 batch_normalization_679 (B  (None, 128, 128, 64)         256
['conv2d_698[0][0]']
 atchNormalization)



 activation_644 (Activation  (None, 128, 128, 64)         0
['batch_normalization_679[0][0
 )                                                                ]'
```

```
]


 conv2d_699 (Conv2D)         (None, 128, 128, 64)         36928
['activation_644[0][0]']


 batch_normalization_680 (B  (None, 128, 128, 64)         256
['conv2d_699[0][0]']
 atchNormalization)



 activation_645 (Activation  (None, 128, 128, 64)         0
['batch_normalization_680[0][0
 )                                                                ]'
]


 conv2d_700 (Conv2D)         (None, 128, 128, 1)          65
['activation_645[0][0]']


==================================================================
============================
Total params: 13996289 (53.39 MB)
Trainable params: 13987073 (53.36 MB)
Non-trainable params: 9216 (36.00 KB)
__________________________________________________________________
_____________________________
```

```
model6.compile(optimizer=Adam(lr=0.001, beta_1=0.9,
beta_2=0.999),loss= 'binary_crossentropy', metrics=[dice_coefficient])
model6.save('Unetpp_model6.h5')

checkpoint = tf.keras.callbacks.ModelCheckpoint("Unetpp_model6.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                   patience=5,
                                                   mode='auto',

restore_best_weights=True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                 factor=0.2,
                                                 patience=2,
                                                 min_lr=1e-6,
                                                 mode='auto',
```

```
                                                verbose=1)
# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]

history6= model6.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model6.save('Unetpp_model6.h5')
```

```
Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.0443 -
dice_coefficient: 0.9759
Epoch 1: val_loss improved from inf to 0.14523, saving model to
Unetpp_model6.h5
394/394 [==============================] - 74s 145ms/step - loss:
0.0443 - dice_coefficient: 0.9759 - val_loss: 0.1452 -
val_dice_coefficient: 0.9884 - lr: 0.0010
Epoch 2/10
394/394 [==============================] - ETA: 0s - loss: 0.0142 -
dice_coefficient: 0.9926
Epoch 2: val_loss improved from 0.14523 to 0.06977, saving model to
Unetpp_model6.h5
394/394 [==============================] - 53s 135ms/step - loss:
0.0142 - dice_coefficient: 0.9926 - val_loss: 0.0698 -
val_dice_coefficient: 0.9913 - lr: 0.0010
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: 0.0171 -
dice_coefficient: 0.9932
Epoch 3: val_loss improved from 0.06977 to 0.06921, saving model to
Unetpp_model6.h5
394/394 [==============================] - 54s 136ms/step - loss:
0.0171 - dice_coefficient: 0.9932 - val_loss: 0.0692 -
val_dice_coefficient: 0.9896 - lr: 0.0010
Epoch 4/10
394/394 [==============================] - ETA: 0s - loss: 0.0096 -
dice_coefficient: 0.9955
Epoch 4: val_loss improved from 0.06921 to 0.00680, saving model to
Unetpp_model6.h5
394/394 [==============================] - 54s 136ms/step - loss:
0.0096 - dice_coefficient: 0.9955 - val_loss: 0.0068 -
val_dice_coefficient: 0.9965 - lr: 0.0010
Epoch 5/10
394/394 [==============================] - ETA: 0s - loss: -5.1873e-05
- dice_coefficient: 0.9951
Epoch 5: val_loss did not improve from 0.00680
394/394 [==============================] - 53s 134ms/step - loss: -
5.1873e-05 - dice_coefficient: 0.9951 - val_loss: 0.1131 -
val_dice_coefficient: 0.9895 - lr: 0.0010
Epoch 6/10
394/394 [==============================] - ETA: 0s - loss: -0.0058 -
```

```
dice_coefficient: 0.9936
Epoch 6: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.

Epoch 6: val_loss did not improve from 0.00680
394/394 [==============================] - 53s 134ms/step - loss: -
0.0058 - dice_coefficient: 0.9936 - val_loss: 0.0432 -
val_dice_coefficient: 0.9946 - lr: 0.0010
Epoch 7/10
394/394 [==============================] - ETA: 0s - loss: 0.0123 -
dice_coefficient: 0.9954
Epoch 7: val_loss did not improve from 0.00680
394/394 [==============================] - 53s 134ms/step - loss:
0.0123 - dice_coefficient: 0.9954 - val_loss: 0.0082 -
val_dice_coefficient: 0.9963 - lr: 2.0000e-04
Epoch 8/10
394/394 [==============================] - ETA: 0s - loss: 0.0045 -
dice_coefficient: 0.9964
Epoch 8: val_loss improved from 0.00680 to -0.00077, saving model to
Unetpp_model6.h5
394/394 [==============================] - 53s 136ms/step - loss:
0.0045 - dice_coefficient: 0.9964 - val_loss: -7.6941e-04 -
val_dice_coefficient: 0.9959 - lr: 2.0000e-04
Epoch 9/10
394/394 [==============================] - ETA: 0s - loss: -0.0173 -
dice_coefficient: 0.9963
Epoch 9: val_loss improved from -0.00077 to -0.07392, saving model to
Unetpp_model6.h5
394/394 [==============================] - 53s 136ms/step - loss: -
0.0173 - dice_coefficient: 0.9963 - val_loss: -0.0739 -
val_dice_coefficient: 0.9935 - lr: 2.0000e-04
Epoch 10/10
394/394 [==============================] - ETA: 0s - loss: -0.0664 -
dice_coefficient: 0.9961
Epoch 10: val_loss improved from -0.07392 to -0.23969, saving model to
Unetpp_model6.h5
394/394 [==============================] - 53s 135ms/step - loss: -
0.0664 - dice_coefficient: 0.9961 - val_loss: -0.2397 -
val_dice_coefficient: 0.9930 - lr: 2.0000e-04
```

```
scores6 = model6.evaluate(x_valid, y_valid)
scores6[1]
```

```
57/57 [==============================] - 6s 79ms/step - loss: -0.2397
- dice_coefficient: 0.9930

0.9930425882339478
```

```
prediction6 = model6.predict(x_test)
```

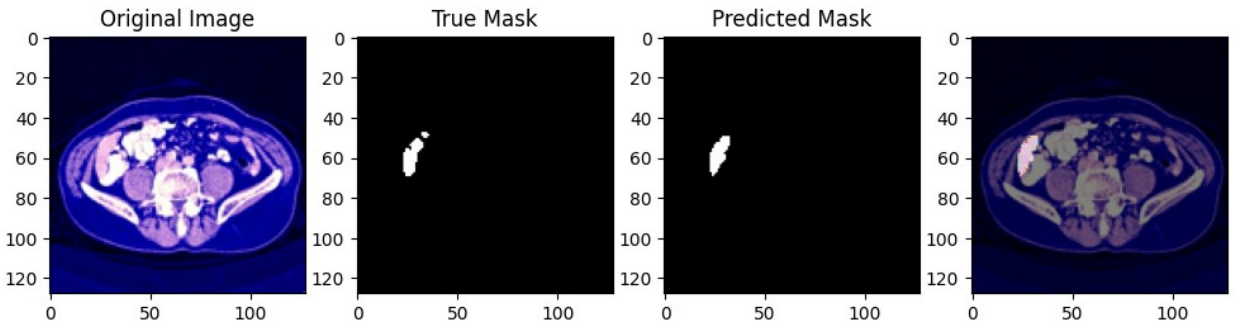
```
test_scores6 = model6.evaluate(x_test, y_test)
test_scores3[1]
```

```
29/29 [==============================] - 3s 95ms/step
29/29 [==============================] - 2s 77ms/step - loss: -0.2557
- dice_coefficient: 0.9926
```

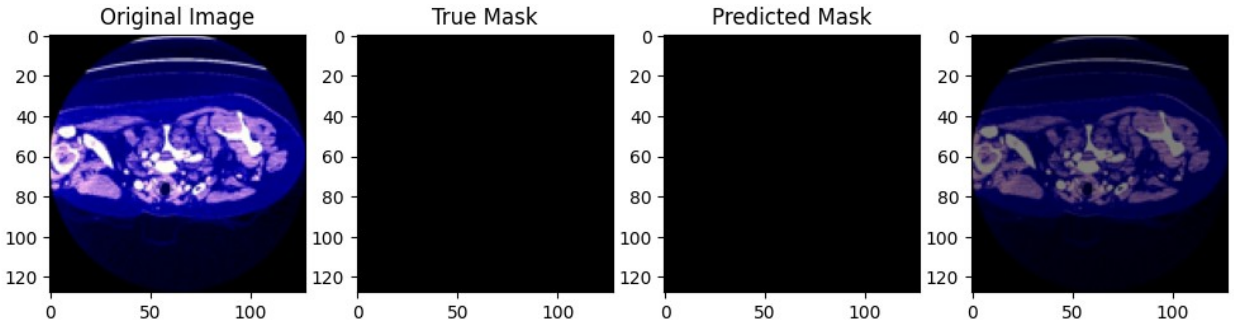
```
0.9914811253547668
```

```
import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 20, replace=False)
for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model3
    predicted_mask = model6.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
    threshold = 0.5  # Adjust this threshold based on your model's
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')
    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
```

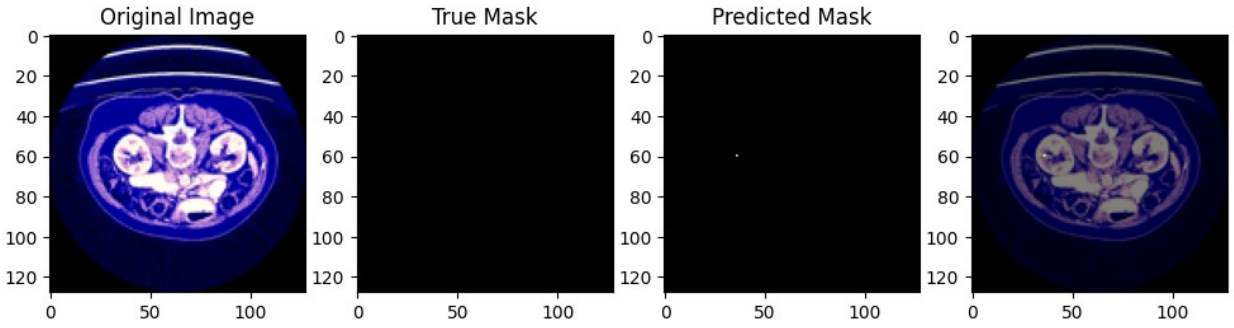
```
'nipy_spectral')
    plt.show()

333
1/1 [==============================] - 0s 24ms/step
```



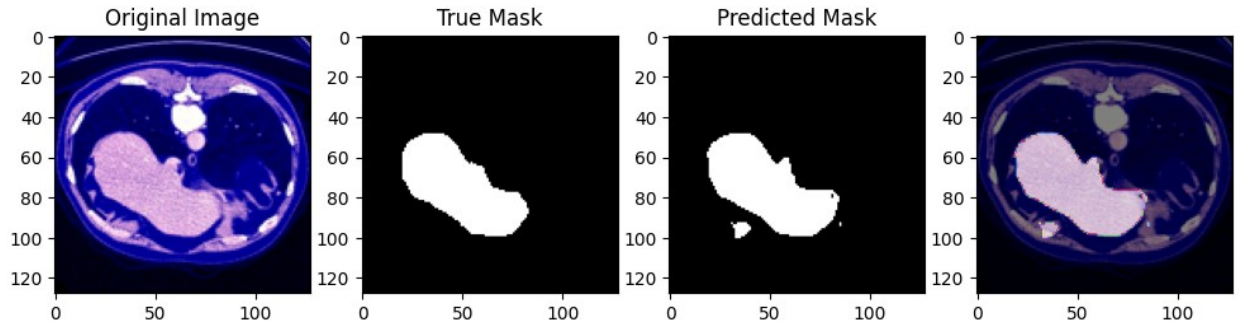

```
812
1/1 [==============================] - 0s 23ms/step
```



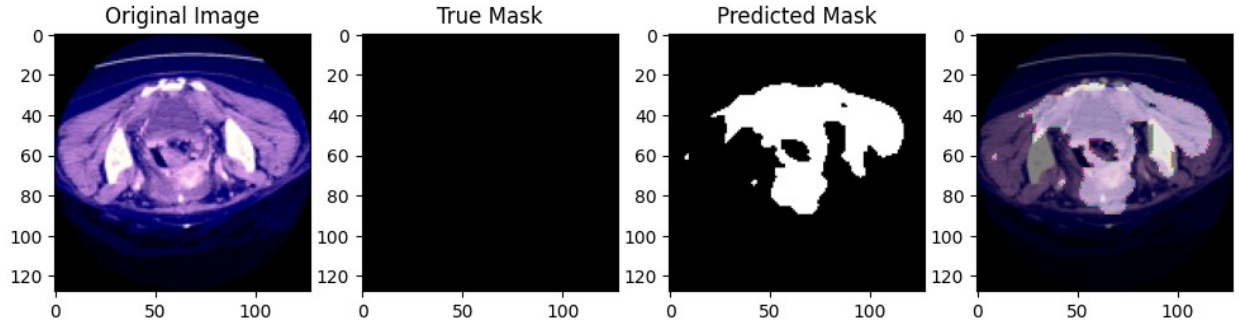

```
745
1/1 [==============================] - 0s 24ms/step
```



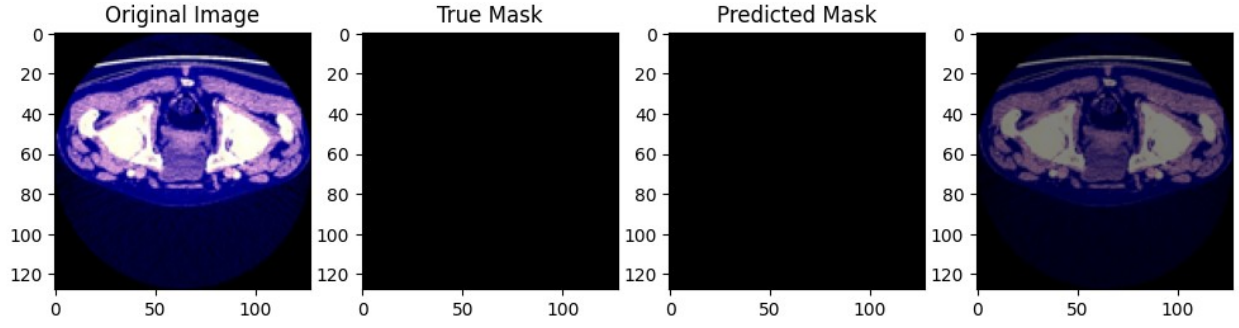

```
245
1/1 [==============================] - 0s 23ms/step
```



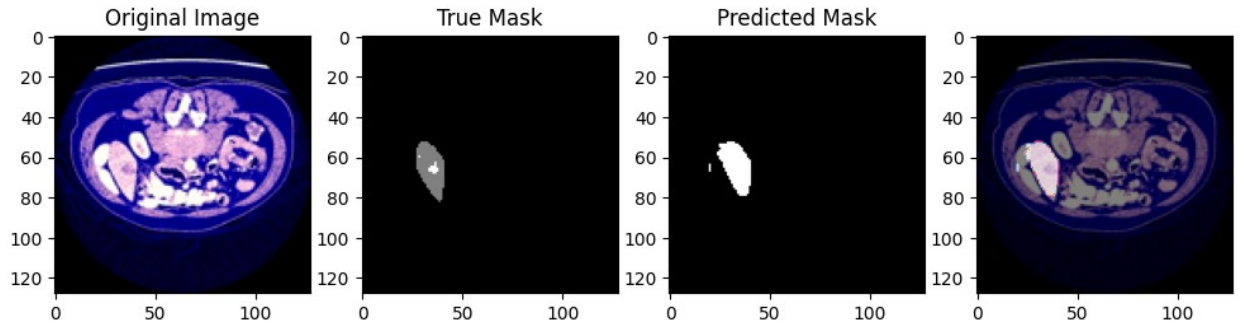

```
40
1/1 [==============================] - 0s 29ms/step
```



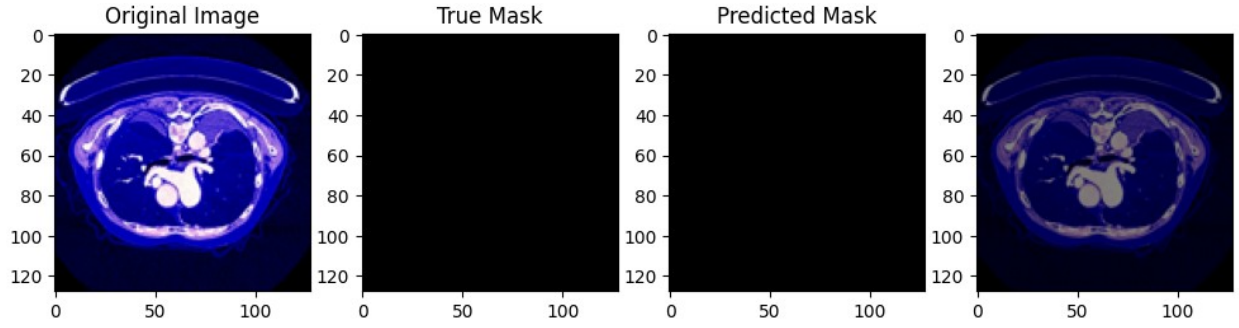

```
427
1/1 [==============================] - 0s 31ms/step
```



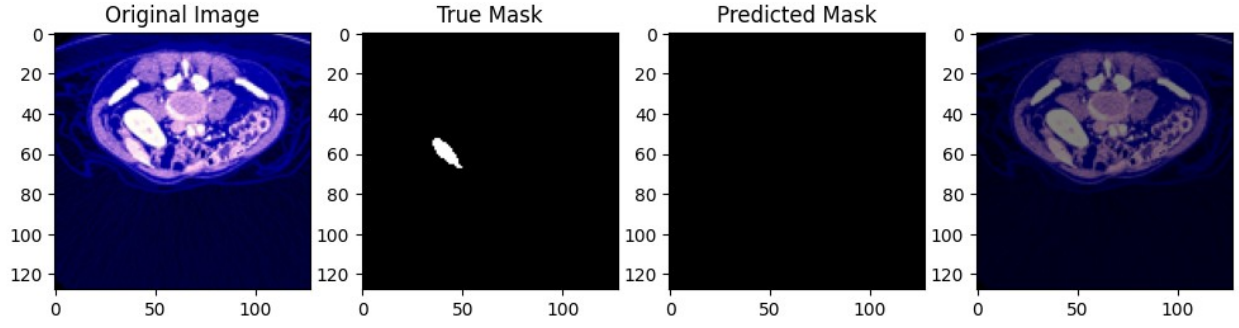

```
303
1/1 [==============================] - 0s 22ms/step
```



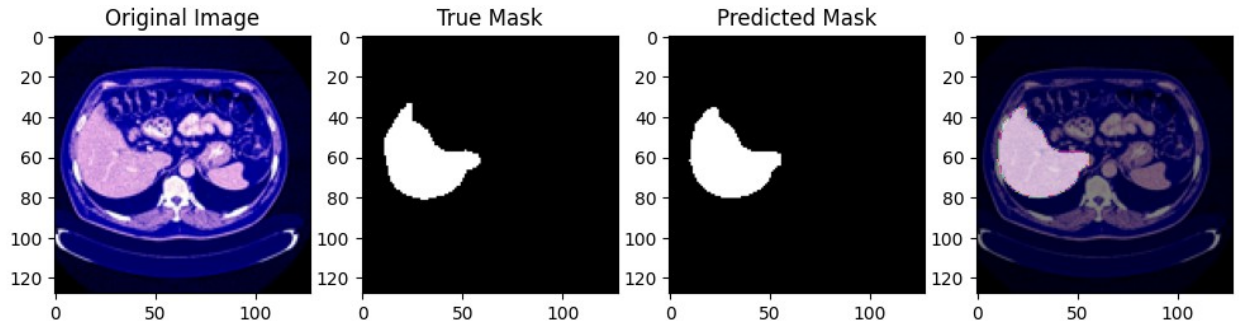

```
588
1/1 [==============================] - 0s 29ms/step
```



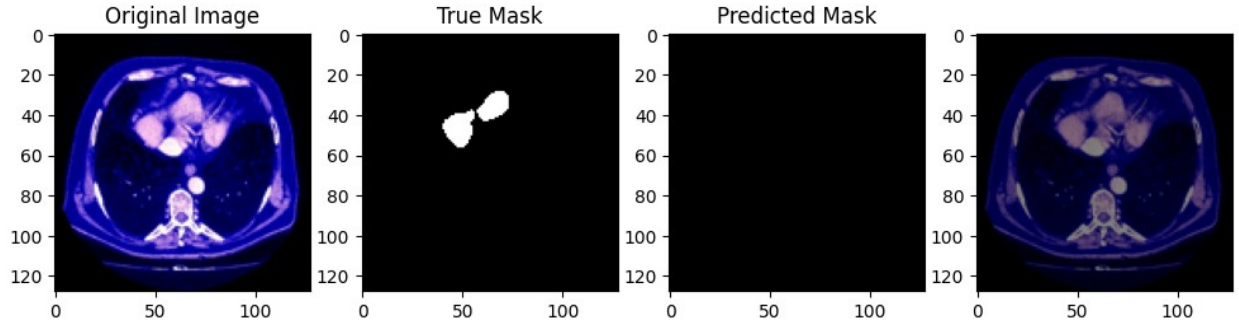

```
210
1/1 [==============================] - 0s 29ms/step
```



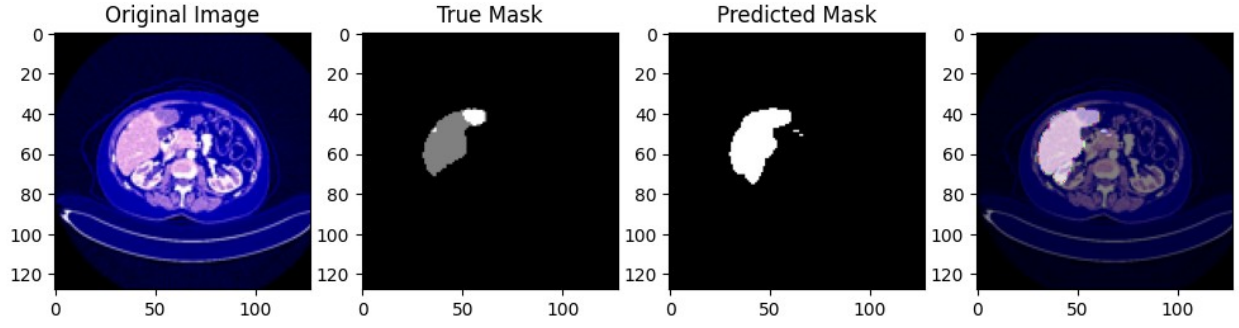

```
137
1/1 [==============================] - 0s 29ms/step
```



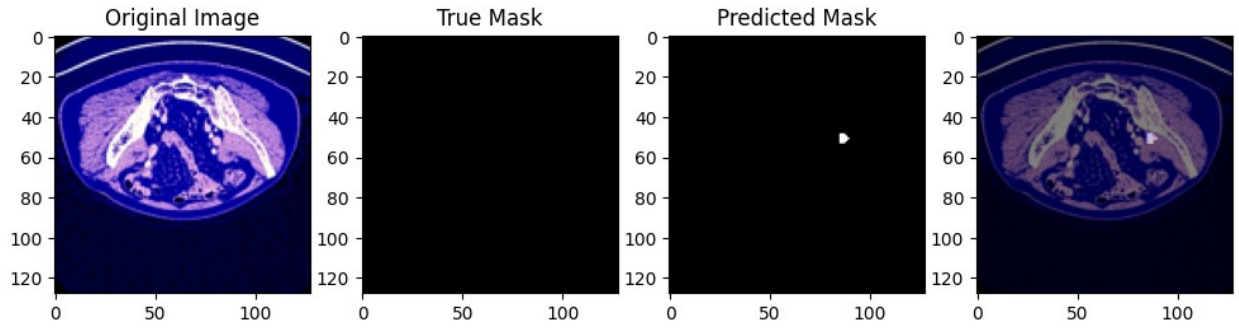

```
138
1/1 [==============================] - 0s 28ms/step
```



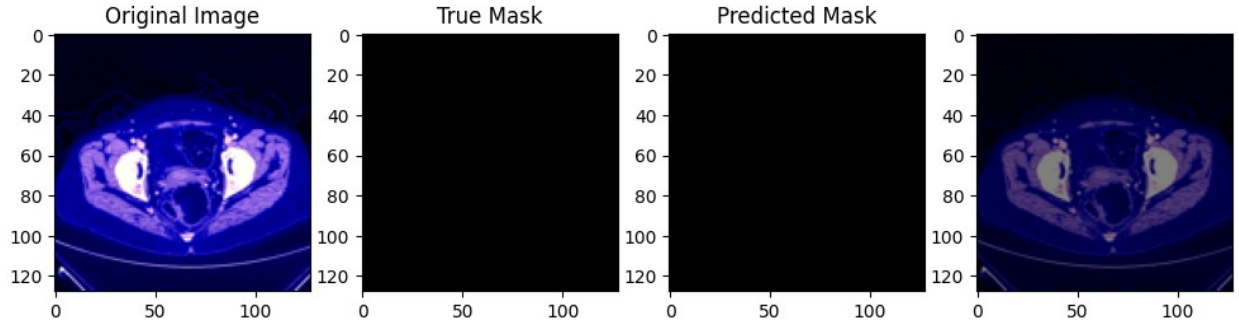

```
261
1/1 [==============================] - 0s 25ms/step
```



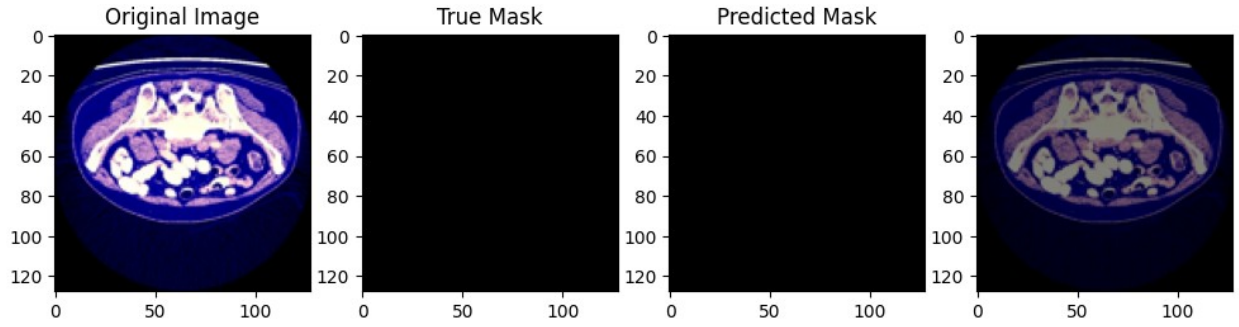

```
352
1/1 [==============================] - 0s 24ms/step
```



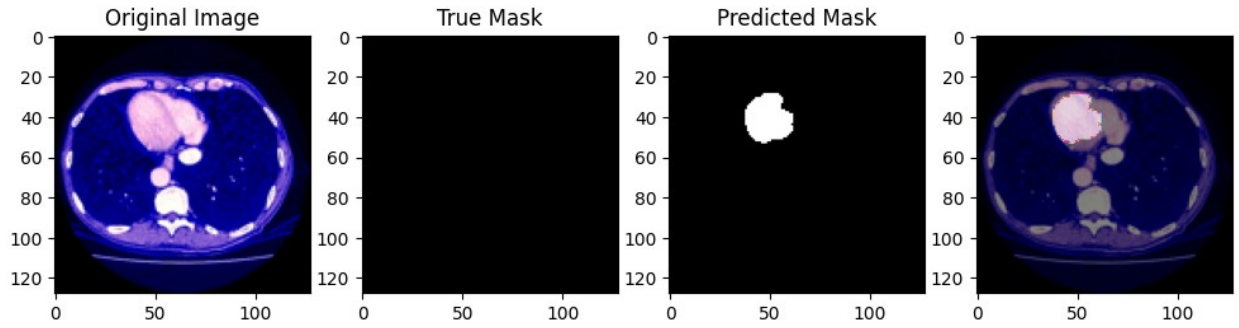

```
360
1/1 [==============================] - 0s 25ms/step
```



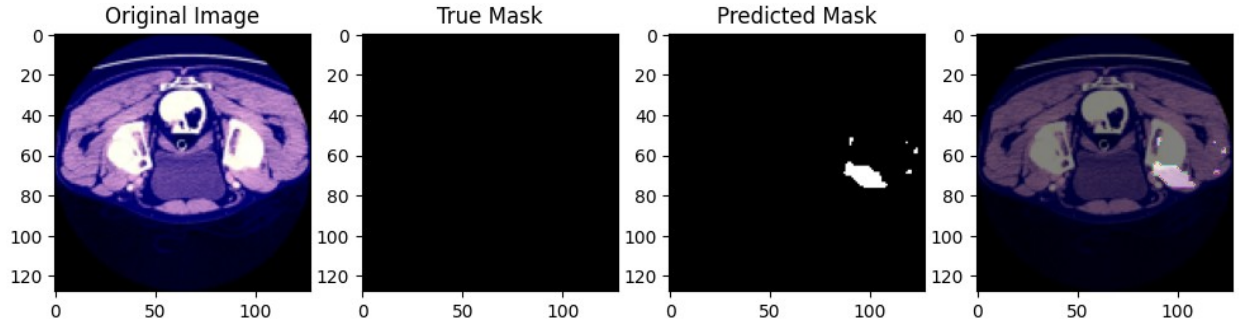

```
347
1/1 [==============================] - 0s 24ms/step
```



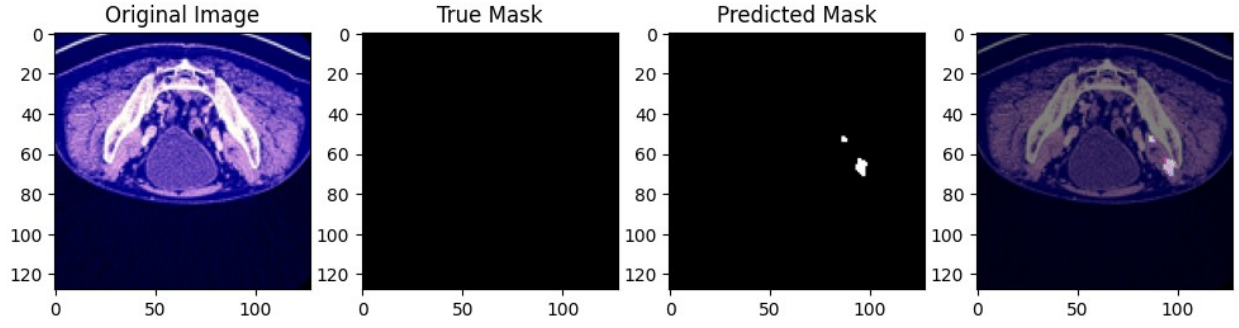

```
479
1/1 [==============================] - 0s 25ms/step
```



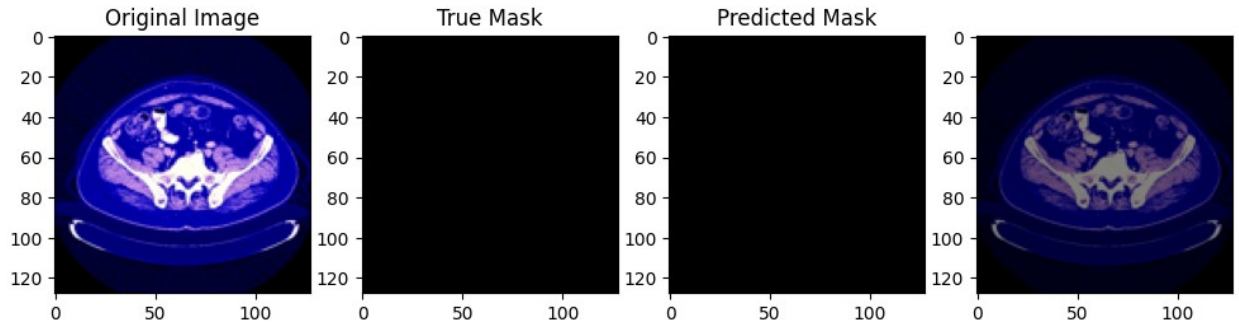

```
307
1/1 [==============================] - 0s 24ms/step
```



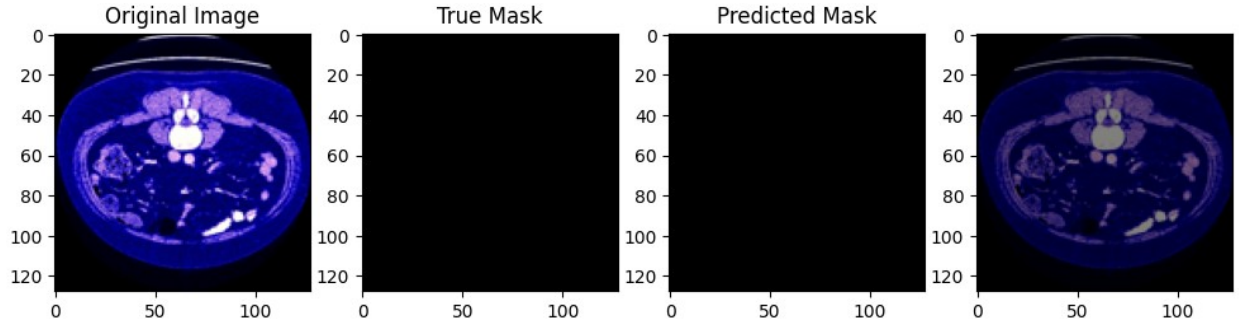

```
658
1/1 [==============================] - 0s 24ms/step
```



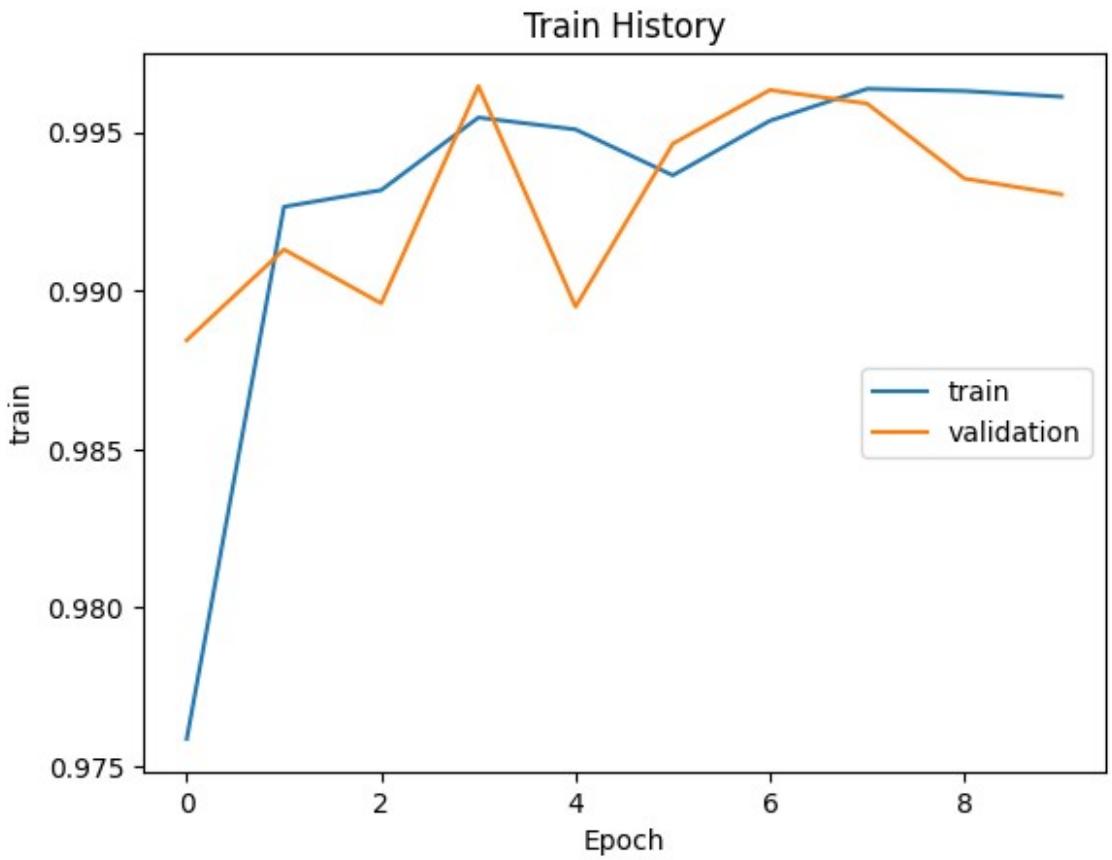

```
895
1/1 [==============================] - 0s 26ms/step
```



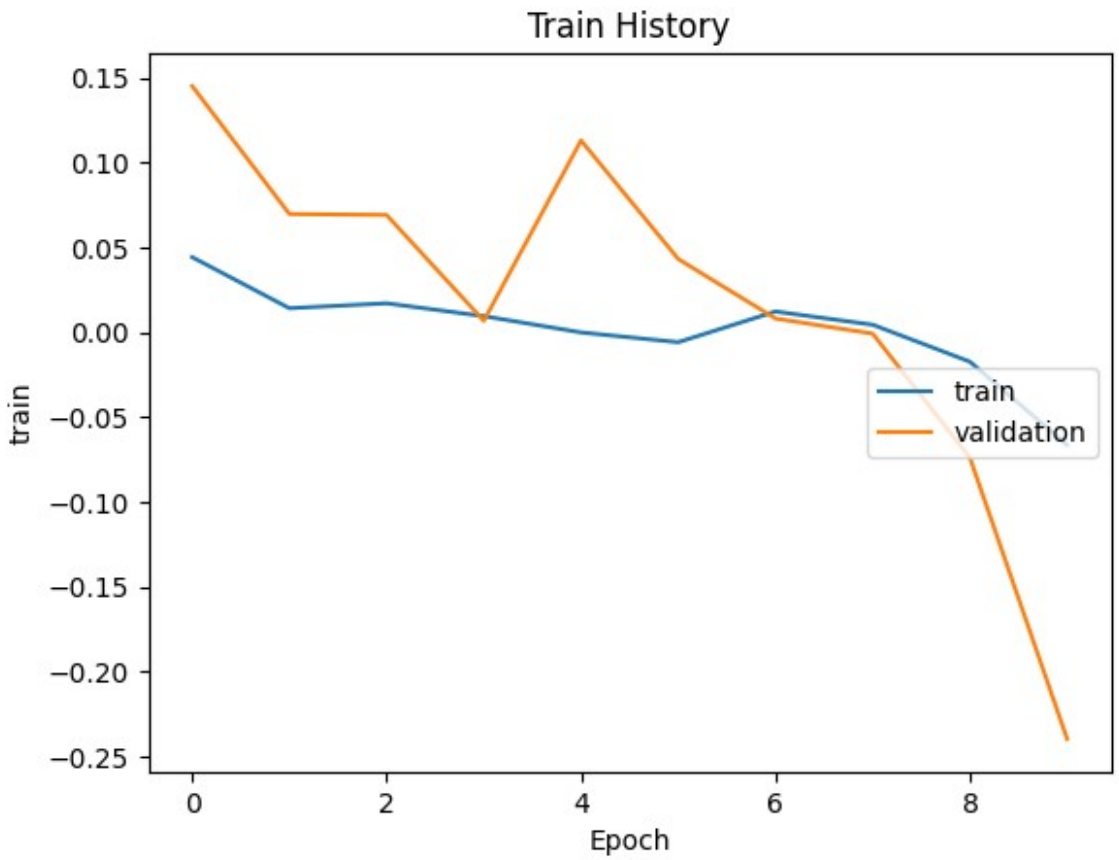

```
111
1/1 [==============================] - 0s 23ms/step
```




```
show_history(history6, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history6, 'loss', 'val_loss')
```




```python
import tensorflow as tf
from tensorflow.keras import layers

def DeepLab():
    inputs = tf.keras.Input(shape=(128, 128, 3))

    # Entry block
    x = layers.Conv2D(32, 3, strides=2, padding="same")(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
    x = layers.Conv2D(64, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)

    # Block 1
    previous_block_activation = x
    for filters in [64, 128]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(filters, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

    # Block 2
    for _ in range(2):
```

```
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(128, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

    # Block 3
    previous_block_activation = x
    for filters in [128, 256]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(filters, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

    # Block 4
    for _ in range(4):
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(256, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

    # Block 5
    previous_block_activation = x
    for filters in [256, 512]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(filters, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

    # Block 6
    for _ in range(4):
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(512, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

    # Block 7
    for _ in range(2):
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(512, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

    # ASPP (Atrous Spatial Pyramid Pooling)
    b4 = layers.Lambda(lambda x: tf.image.resize(x, (tf.shape(x)[1],
tf.shape(x)[2])))(previous_block_activation)
    b4 = layers.Conv2D(512, 1, padding="same")(b4)
    b4 = layers.BatchNormalization()(b4)

    b0 = layers.Conv2D(512, 1, padding="same")(x)
    b0 = layers.BatchNormalization()(b0)

    # Concatenate ASPP branches & reduce filters
    x = layers.Concatenate()([b4, b0])
    x = layers.Conv2D(1024, 1, padding="same")(x)
    x = layers.BatchNormalization()(x)
```

```
    x = layers.ReLU()(x)

    # DeepLab head
    x = layers.Conv2D(1, 1, activation="sigmoid", padding="same")(x)
    outputs = layers.UpSampling2D((2, 2))(x)  # Adjusted upsampling
factor to match label size

    model = tf.keras.Model(inputs, outputs)
    return model

# Instantiate the model
model7 = DeepLab()

# Compile the model
model7.compile(optimizer='Adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Display the model summary
model7.summary()
```

```
Model: "model_17"
__________________________________________________________________________
_____
 Layer (type)                Output Shape              Param #
Connected to
==========================================================================
=============================
 input_18 (InputLayer)       [(None, 128, 128, 3)]     0         []



 conv2d_719 (Conv2D)         (None, 64, 64, 32)        896
['input_18[0][0]']


 batch_normalization_750 (B  (None, 64, 64, 32)        128
['conv2d_719[0][0]']
 atchNormalization)



 re_lu_9 (ReLU)              (None, 64, 64, 32)        0
['batch_normalization_750[0][0
                                                                  ]'
]


 conv2d_720 (Conv2D)         (None, 64, 64, 64)        18496
['re_lu_9[0][0]']
```

```
 batch_normalization_751 (B  (None, 64, 64, 64)           256
['conv2d_720[0][0]']
 atchNormalization)



 re_lu_10 (ReLU)             (None, 64, 64, 64)           0
['batch_normalization_751[0][0
                                                                 ]'
]


 activation_700 (Activation  (None, 64, 64, 64)           0
['re_lu_10[0][0]']
 )



 separable_conv2d_54 (Separ  (None, 64, 64, 64)           4736
['activation_700[0][0]']
 ableConv2D)



 batch_normalization_752 (B  (None, 64, 64, 64)           256
['separable_conv2d_54[0][0]']
 atchNormalization)



 activation_701 (Activation  (None, 64, 64, 64)           0
['batch_normalization_752[0][0
 )                                                       ]'
]


 separable_conv2d_55 (Separ  (None, 64, 64, 128)          8896
['activation_701[0][0]']
 ableConv2D)



 batch_normalization_753 (B  (None, 64, 64, 128)          512
['separable_conv2d_55[0][0]']
 atchNormalization)



 activation_702 (Activation  (None, 64, 64, 128)          0
['batch_normalization_753[0][0
```

```
 )                                                                  ]'
]


 separable_conv2d_56 (Separ  (None, 64, 64, 128)          17664
['activation_702[0][0]']
 ableConv2D)



 batch_normalization_754 (B  (None, 64, 64, 128)          512
['separable_conv2d_56[0][0]']
 atchNormalization)



 activation_703 (Activation  (None, 64, 64, 128)          0
['batch_normalization_754[0][0
 )                                                                  ]'
]


 separable_conv2d_57 (Separ  (None, 64, 64, 128)          17664
['activation_703[0][0]']
 ableConv2D)



 batch_normalization_755 (B  (None, 64, 64, 128)          512
['separable_conv2d_57[0][0]']
 atchNormalization)



 activation_704 (Activation  (None, 64, 64, 128)          0
['batch_normalization_755[0][0
 )                                                                  ]'
]


 separable_conv2d_58 (Separ  (None, 64, 64, 128)          17664
['activation_704[0][0]']
 ableConv2D)



 batch_normalization_756 (B  (None, 64, 64, 128)          512
['separable_conv2d_58[0][0]']
 atchNormalization)
```

```
 activation_705 (Activation  (None, 64, 64, 128)          0
['batch_normalization_756[0][0
 )                                                                ]'
]


 separable_conv2d_59 (Separ  (None, 64, 64, 256)          34176
['activation_705[0][0]']
 ableConv2D)



 batch_normalization_757 (B  (None, 64, 64, 256)          1024
['separable_conv2d_59[0][0]']
 atchNormalization)



 activation_706 (Activation  (None, 64, 64, 256)          0
['batch_normalization_757[0][0
 )                                                                ]'
]


 separable_conv2d_60 (Separ  (None, 64, 64, 256)          68096
['activation_706[0][0]']
 ableConv2D)



 batch_normalization_758 (B  (None, 64, 64, 256)          1024
['separable_conv2d_60[0][0]']
 atchNormalization)



 activation_707 (Activation  (None, 64, 64, 256)          0
['batch_normalization_758[0][0
 )                                                                ]'
]


 separable_conv2d_61 (Separ  (None, 64, 64, 256)          68096
['activation_707[0][0]']
 ableConv2D)



 batch_normalization_759 (B  (None, 64, 64, 256)          1024
['separable_conv2d_61[0][0]']
```

```
 atchNormalization)


 activation_708 (Activation  (None, 64, 64, 256)          0
['batch_normalization_759[0][0
 )                                                                ]'
]


 separable_conv2d_62 (Separ  (None, 64, 64, 256)          68096
['activation_708[0][0]']
 ableConv2D)


 batch_normalization_760 (B  (None, 64, 64, 256)          1024
['separable_conv2d_62[0][0]']
 atchNormalization)


 activation_709 (Activation  (None, 64, 64, 256)          0
['batch_normalization_760[0][0
 )                                                                ]'
]


 separable_conv2d_63 (Separ  (None, 64, 64, 256)          68096
['activation_709[0][0]']
 ableConv2D)


 batch_normalization_761 (B  (None, 64, 64, 256)          1024
['separable_conv2d_63[0][0]']
 atchNormalization)


 activation_710 (Activation  (None, 64, 64, 256)          0
['batch_normalization_761[0][0
 )                                                                ]'
]


 separable_conv2d_64 (Separ  (None, 64, 64, 256)          68096
['activation_710[0][0]']
 ableConv2D)
```

```
 batch_normalization_762 (B  (None, 64, 64, 256)          1024
['separable_conv2d_64[0][0]']
 atchNormalization)



 activation_711 (Activation  (None, 64, 64, 256)          0
['batch_normalization_762[0][0
 )                                                                ]'
]


 separable_conv2d_65 (Separ  (None, 64, 64, 512)          133888
['activation_711[0][0]']
 ableConv2D)



 batch_normalization_763 (B  (None, 64, 64, 512)          2048
['separable_conv2d_65[0][0]']
 atchNormalization)



 activation_712 (Activation  (None, 64, 64, 512)          0
['batch_normalization_763[0][0
 )                                                                ]'
]


 separable_conv2d_66 (Separ  (None, 64, 64, 512)          267264
['activation_712[0][0]']
 ableConv2D)



 batch_normalization_764 (B  (None, 64, 64, 512)          2048
['separable_conv2d_66[0][0]']
 atchNormalization)



 activation_713 (Activation  (None, 64, 64, 512)          0
['batch_normalization_764[0][0
 )                                                                ]'
]


 separable_conv2d_67 (Separ  (None, 64, 64, 512)          267264
['activation_713[0][0]']
```

```
 ableConv2D)


 batch_normalization_765 (B  (None, 64, 64, 512)          2048
['separable_conv2d_67[0][0]']
 atchNormalization)


 activation_714 (Activation  (None, 64, 64, 512)          0
['batch_normalization_765[0][0
 )                                                                ]'
]


 separable_conv2d_68 (Separ  (None, 64, 64, 512)          267264
['activation_714[0][0]']
 ableConv2D)


 batch_normalization_766 (B  (None, 64, 64, 512)          2048
['separable_conv2d_68[0][0]']
 atchNormalization)


 activation_715 (Activation  (None, 64, 64, 512)          0
['batch_normalization_766[0][0
 )                                                                ]'
]


 separable_conv2d_69 (Separ  (None, 64, 64, 512)          267264
['activation_715[0][0]']
 ableConv2D)


 batch_normalization_767 (B  (None, 64, 64, 512)          2048
['separable_conv2d_69[0][0]']
 atchNormalization)


 activation_716 (Activation  (None, 64, 64, 512)          0
['batch_normalization_767[0][0
 )                                                                ]'
]
```

```
 separable_conv2d_70 (Separ  (None, 64, 64, 512)          267264
['activation_716[0][0]']
 ableConv2D)


 batch_normalization_768 (B  (None, 64, 64, 512)          2048
['separable_conv2d_70[0][0]']
 atchNormalization)


 activation_717 (Activation  (None, 64, 64, 512)          0
['batch_normalization_768[0][0
 )                                                                ]'
]

 separable_conv2d_71 (Separ  (None, 64, 64, 512)          267264
['activation_717[0][0]']
 ableConv2D)


 lambda_3 (Lambda)           (None, 64, 64, 256)          0
['batch_normalization_761[0][0
                                                                  ]'
]

 batch_normalization_769 (B  (None, 64, 64, 512)          2048
['separable_conv2d_71[0][0]']
 atchNormalization)


 conv2d_721 (Conv2D)         (None, 64, 64, 512)          131584
['lambda_3[0][0]']

 conv2d_722 (Conv2D)         (None, 64, 64, 512)          262656
['batch_normalization_769[0][0
                                                                  ]'
]

 batch_normalization_770 (B  (None, 64, 64, 512)          2048
['conv2d_721[0][0]']
 atchNormalization)
```

```
 batch_normalization_771 (B  (None, 64, 64, 512)          2048
['conv2d_722[0][0]']
 atchNormalization)



 concatenate_538 (Concatena  (None, 64, 64, 1024)         0
['batch_normalization_770[0][0
te)                                                                ]',


'batch_normalization_771[0][0
                                                                    ]'
]


 conv2d_723 (Conv2D)         (None, 64, 64, 1024)         1049600
['concatenate_538[0][0]']


 batch_normalization_772 (B  (None, 64, 64, 1024)         4096
['conv2d_723[0][0]']
 atchNormalization)



 re_lu_11 (ReLU)             (None, 64, 64, 1024)         0
['batch_normalization_772[0][0
                                                                    ]'
]


 conv2d_724 (Conv2D)         (None, 64, 64, 1)            1025
['re_lu_11[0][0]']


 up_sampling2d_35 (UpSampli  (None, 128, 128, 1)          0
['conv2d_724[0][0]']
 ng2D)



==========================================================================
============================
Total params: 3674369 (14.02 MB)
Trainable params: 3658689 (13.96 MB)
Non-trainable params: 15680 (61.25 KB)
```

```
_____
_____

model7.compile(optimizer=Adam(lr=0.001, beta_1=0.9,
beta_2=0.999),loss= 'binary_crossentropy', metrics=[dice_coefficient])
model7.save('Deeplab_model7.h5')

checkpoint = tf.keras.callbacks.ModelCheckpoint("Deeplab_model7.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                   patience=5,
                                                   mode='auto',

restore_best_weights=True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                 factor=0.2,
                                                 patience=2,
                                                 min_lr=1e-6,
                                                 mode='auto',
                                                 verbose=1)
# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]

history7= model7.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model7.save('Deeplab_model7.h5')

Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.0402 -
dice_coefficient: 0.9878
Epoch 1: val_loss improved from inf to 0.13920, saving model to
Deeplab_model7.h5
394/394 [==============================] - 188s 420ms/step - loss:
0.0402 - dice_coefficient: 0.9878 - val_loss: 0.1392 -
val_dice_coefficient: 0.9166 - lr: 0.0010
Epoch 2/10
394/394 [==============================] - ETA: 0s - loss: 0.0094 -
dice_coefficient: 0.9951
Epoch 2: val_loss improved from 0.13920 to 0.03897, saving model to
Deeplab_model7.h5
394/394 [==============================] - 161s 409ms/step - loss:
0.0094 - dice_coefficient: 0.9951 - val_loss: 0.0390 -
val_dice_coefficient: 0.9845 - lr: 0.0010
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: 0.0028 -
dice_coefficient: 0.9961
```

```
Epoch 3: val_loss improved from 0.03897 to 0.00500, saving model to
Deeplab_model7.h5
394/394 [==============================] - 161s 409ms/step - loss:
0.0028 - dice_coefficient: 0.9961 - val_loss: 0.0050 -
val_dice_coefficient: 0.9969 - lr: 0.0010
Epoch 4/10
394/394 [==============================] - ETA: 0s - loss: 2.9106e-04
- dice_coefficient: 0.9965
Epoch 4: val_loss did not improve from 0.00500
394/394 [==============================] - 161s 408ms/step - loss:
2.9106e-04 - dice_coefficient: 0.9965 - val_loss: 0.0171 -
val_dice_coefficient: 0.9907 - lr: 0.0010
Epoch 5/10
394/394 [==============================] - ETA: 0s - loss: -0.0025 -
dice_coefficient: 0.9969
Epoch 5: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.

Epoch 5: val_loss did not improve from 0.00500
394/394 [==============================] - 161s 408ms/step - loss: -
0.0025 - dice_coefficient: 0.9969 - val_loss: 0.0194 -
val_dice_coefficient: 0.9964 - lr: 0.0010
Epoch 6/10
394/394 [==============================] - ETA: 0s - loss: -0.0077 -
dice_coefficient: 0.9977
Epoch 6: val_loss improved from 0.00500 to -0.00671, saving model to
Deeplab_model7.h5
394/394 [==============================] - 161s 409ms/step - loss: -
0.0077 - dice_coefficient: 0.9977 - val_loss: -0.0067 -
val_dice_coefficient: 0.9975 - lr: 2.0000e-04
Epoch 7/10
394/394 [==============================] - ETA: 0s - loss: -0.0094 -
dice_coefficient: 0.9981
Epoch 7: val_loss improved from -0.00671 to -0.00755, saving model to
Deeplab_model7.h5
394/394 [==============================] - 161s 409ms/step - loss: -
0.0094 - dice_coefficient: 0.9981 - val_loss: -0.0075 -
val_dice_coefficient: 0.9980 - lr: 2.0000e-04
Epoch 8/10
394/394 [==============================] - ETA: 0s - loss: -0.0101 -
dice_coefficient: 0.9982
Epoch 8: val_loss improved from -0.00755 to -0.00778, saving model to
Deeplab_model7.h5
394/394 [==============================] - 161s 409ms/step - loss: -
0.0101 - dice_coefficient: 0.9982 - val_loss: -0.0078 -
val_dice_coefficient: 0.9982 - lr: 2.0000e-04
Epoch 9/10
394/394 [==============================] - ETA: 0s - loss: -0.0108 -
dice_coefficient: 0.9983
```

```
Epoch 9: val_loss improved from -0.00778 to -0.00802, saving model to
Deeplab_model7.h5
394/394 [==============================] - 168s 426ms/step - loss: -
0.0108 - dice_coefficient: 0.9983 - val_loss: -0.0080 -
val_dice_coefficient: 0.9983 - lr: 2.0000e-04
Epoch 10/10
394/394 [==============================] - ETA: 0s - loss: -0.0104 -
dice_coefficient: 0.9983
Epoch 10: val_loss did not improve from -0.00802
394/394 [==============================] - 161s 408ms/step - loss: -
0.0104 - dice_coefficient: 0.9983 - val_loss: -0.0028 -
val_dice_coefficient: 0.9982 - lr: 2.0000e-04
```

```
scores7 = model7.evaluate(x_valid, y_valid)
scores7[1]
```

```
57/57 [==============================] - 15s 241ms/step - loss: -
0.0028 - dice_coefficient: 0.9982
```

```
0.9981849789619446
```

```
prediction7 = model7.predict(x_test)
```

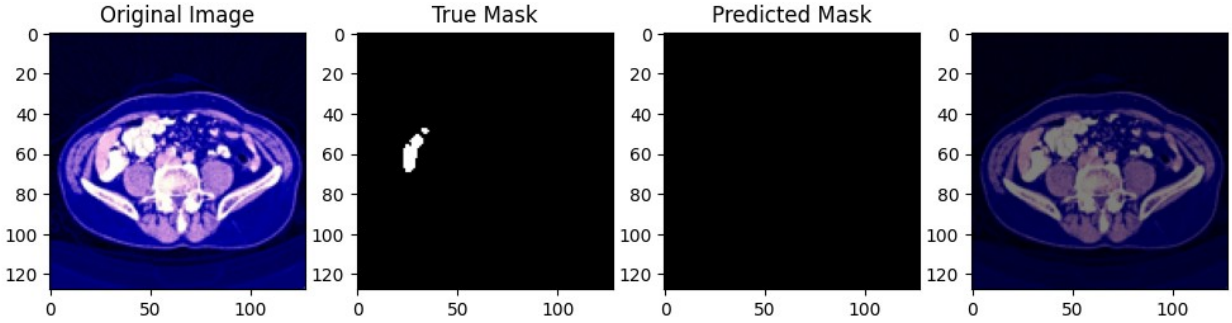
```
test_scores7 = model7.evaluate(x_test, y_test)
test_scores7[1]
```

```
29/29 [==============================] - 8s 251ms/step
29/29 [==============================] - 7s 236ms/step - loss: -0.0054
- dice_coefficient: 0.9982
```

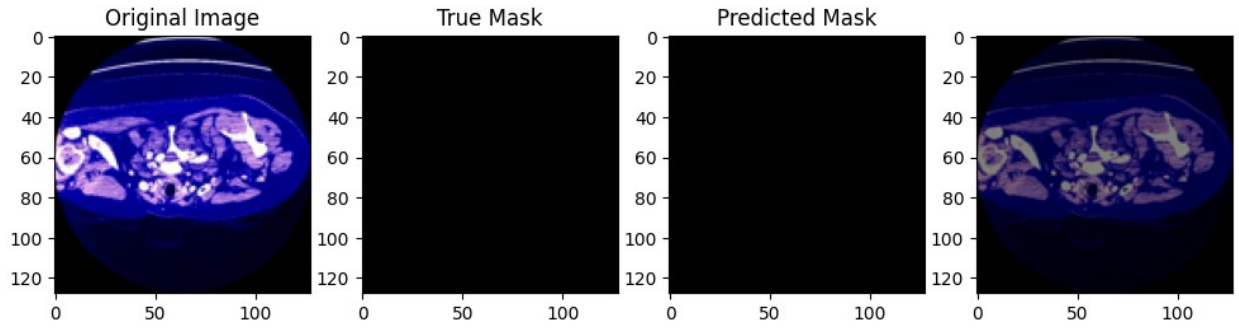
```
0.9981764554977417
```

```
import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 20, replace=False)
for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model3
    predicted_mask = model7.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
```

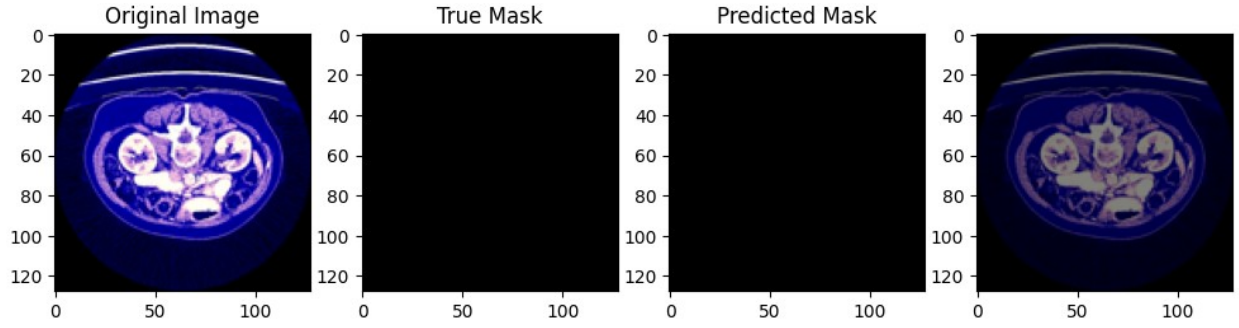
```
    threshold = 0.5  # Adjust this threshold based on your model's
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')
    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
'nipy_spectral')
    plt.show()

333
1/1 [==============================] - 0s 206ms/step
```



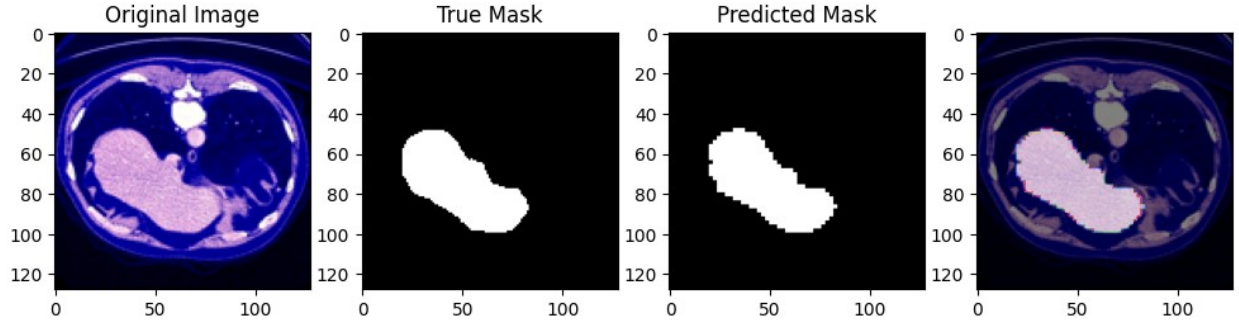

```
812
1/1 [==============================] - 0s 29ms/step
```



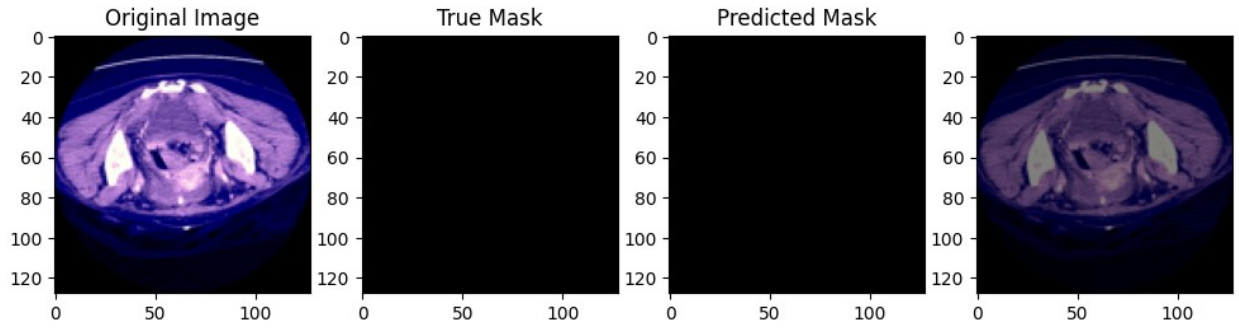

```
745
1/1 [==============================] - 0s 28ms/step
```



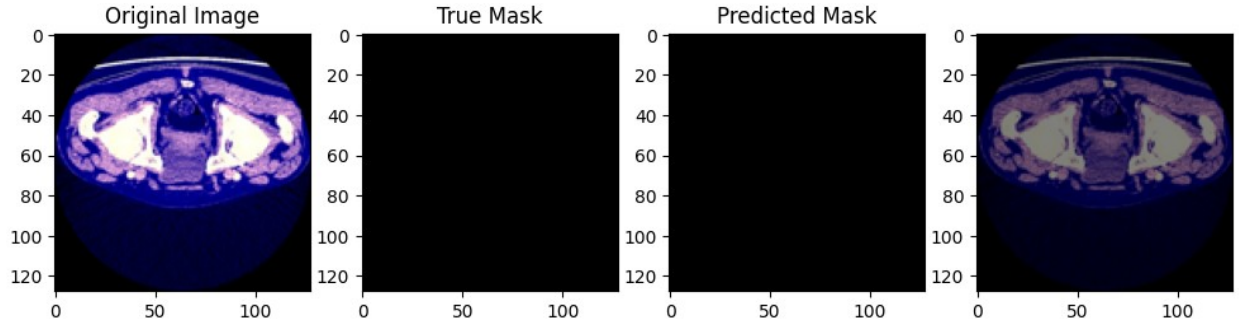

```
245
1/1 [==============================] - 0s 28ms/step
```



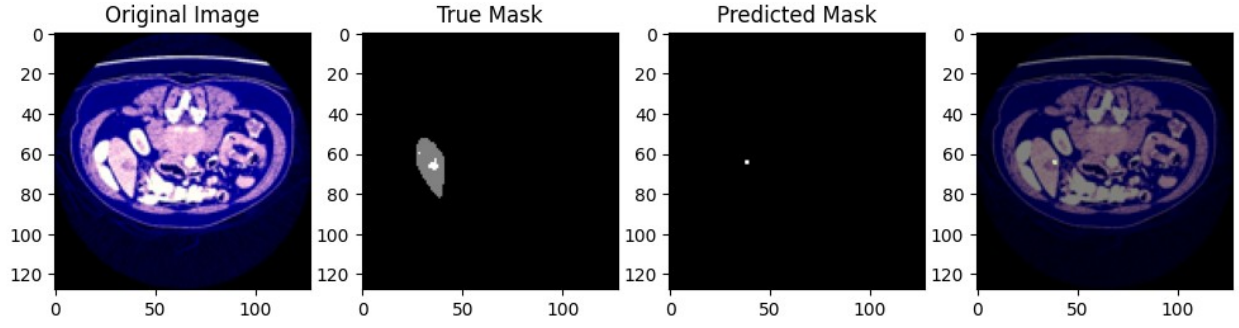

```
40
1/1 [==============================] - 0s 28ms/step
```



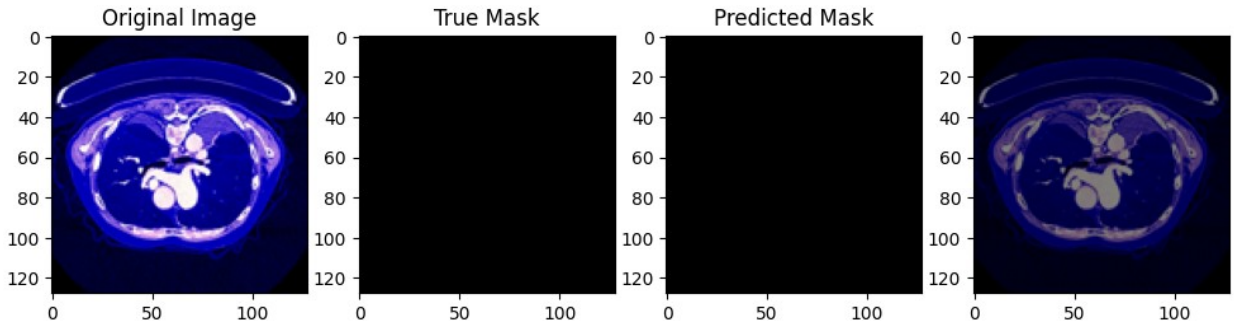

```
427
1/1 [==============================] - 0s 26ms/step
```



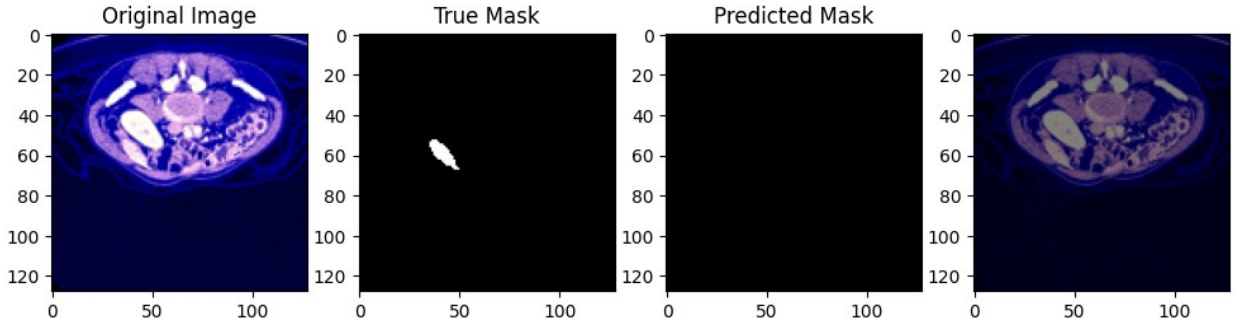

```
303
1/1 [==============================] - 0s 25ms/step
```



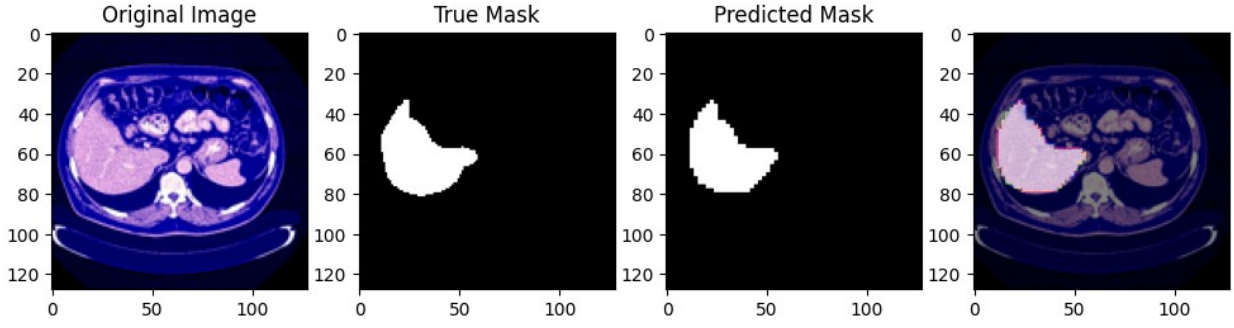

```
588
1/1 [==============================] - 0s 25ms/step
```



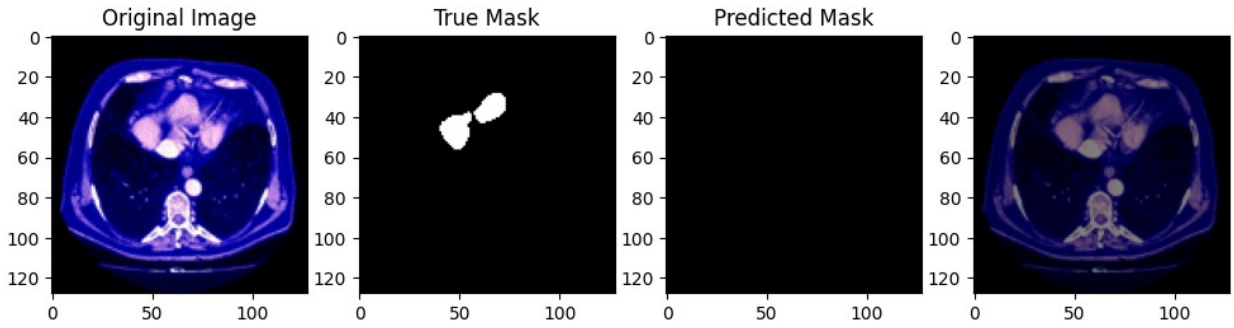

```
210
1/1 [==============================] - 0s 29ms/step
```



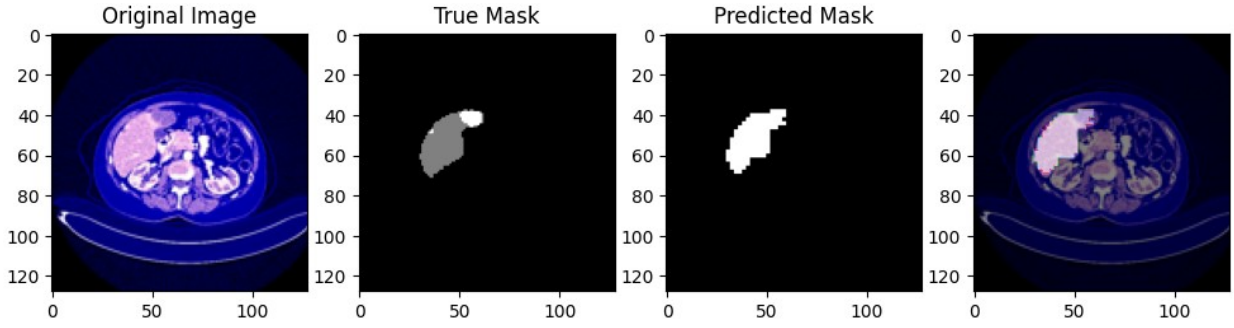

```
137
1/1 [==============================] - 0s 25ms/step
```



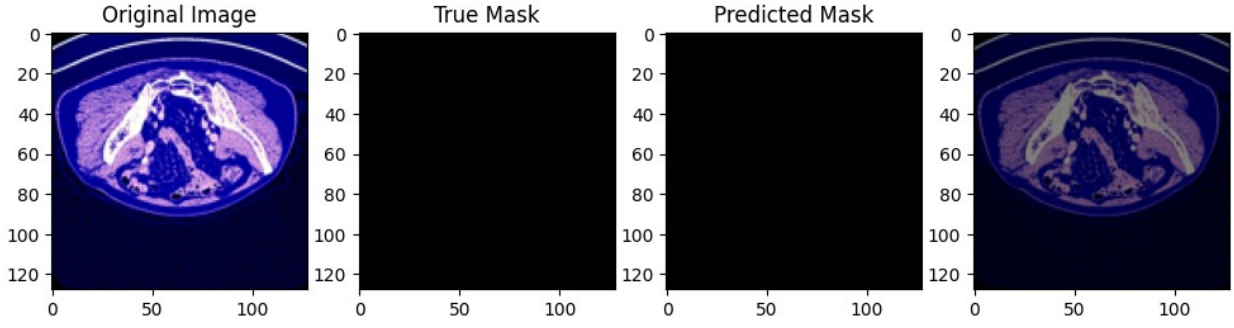

```
138
1/1 [==============================] - 0s 24ms/step
```



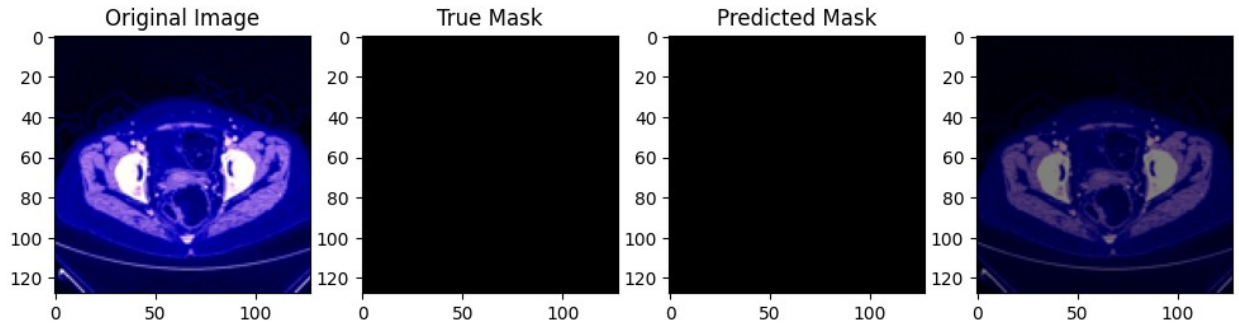

```
261
1/1 [==============================] - 0s 25ms/step
```



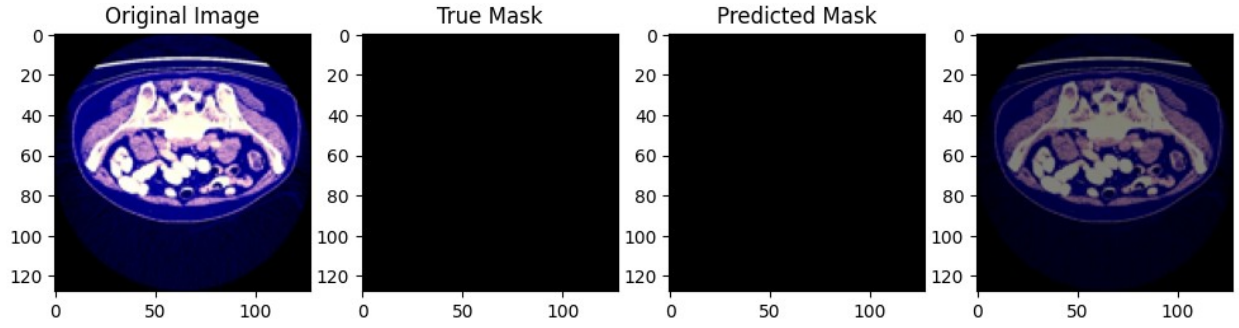

```
352
1/1 [==============================] - 0s 24ms/step
```



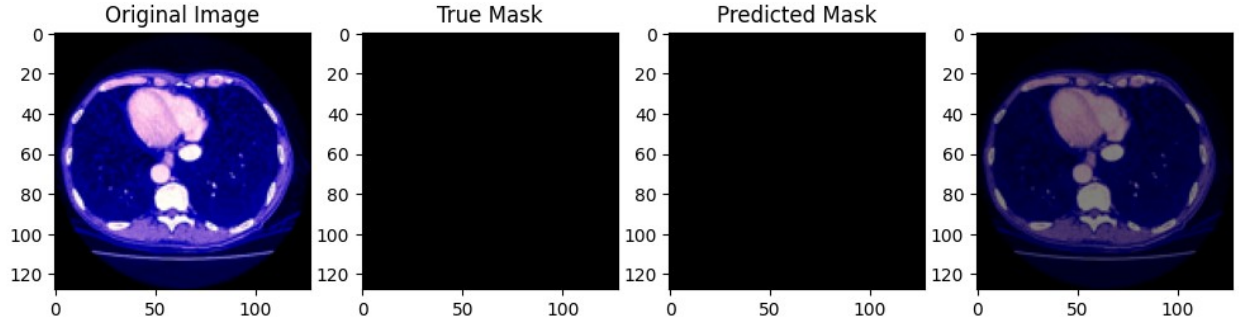

```
360
1/1 [==============================] - 0s 25ms/step
```



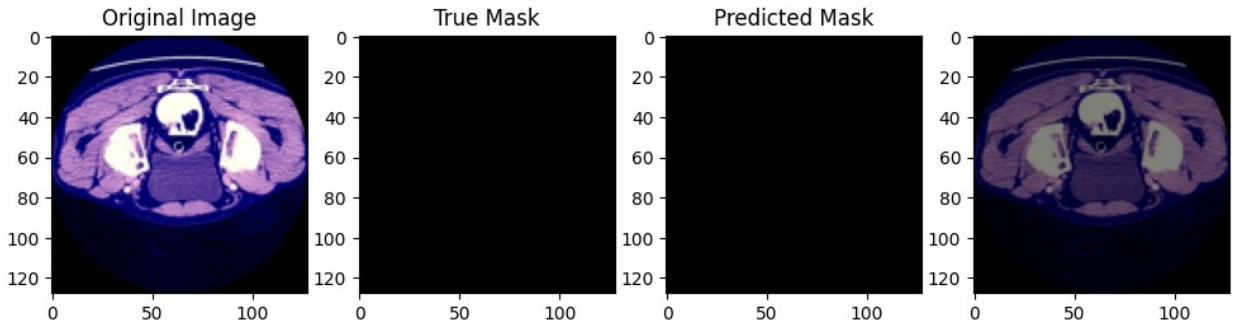

```
347
1/1 [==============================] - 0s 24ms/step
```



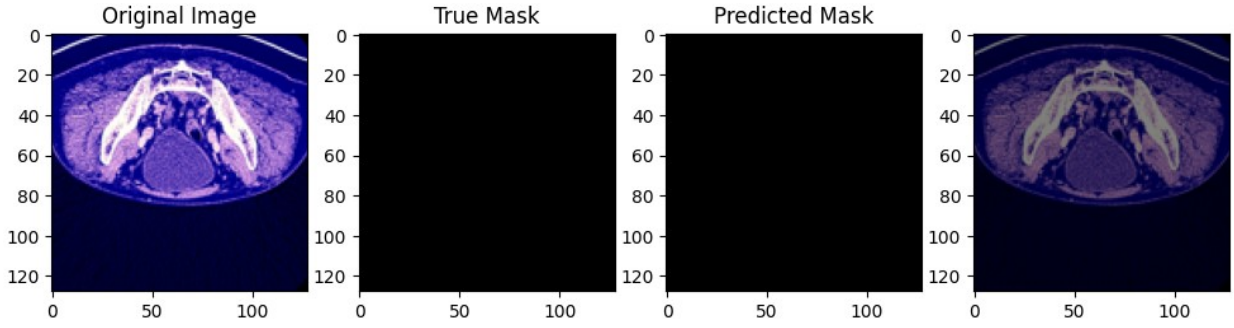

```
479
1/1 [==============================] - 0s 26ms/step
```



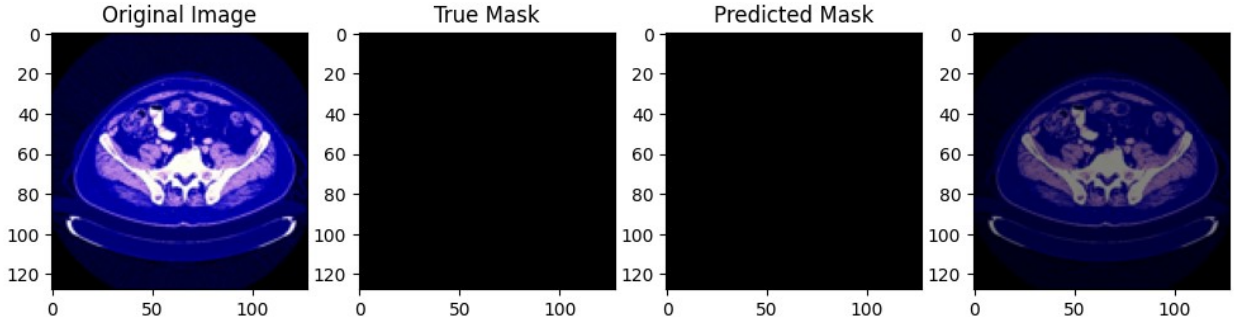

```
307
1/1 [==============================] - 0s 24ms/step
```



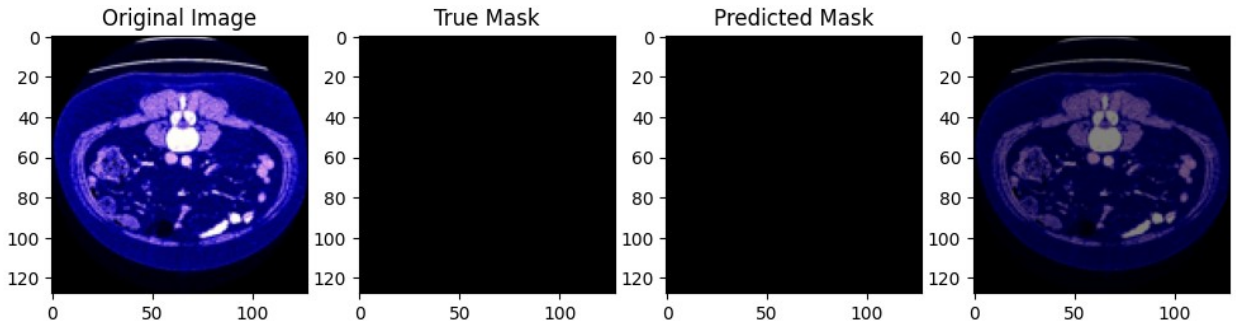

```
658
1/1 [==============================] - 0s 23ms/step
```



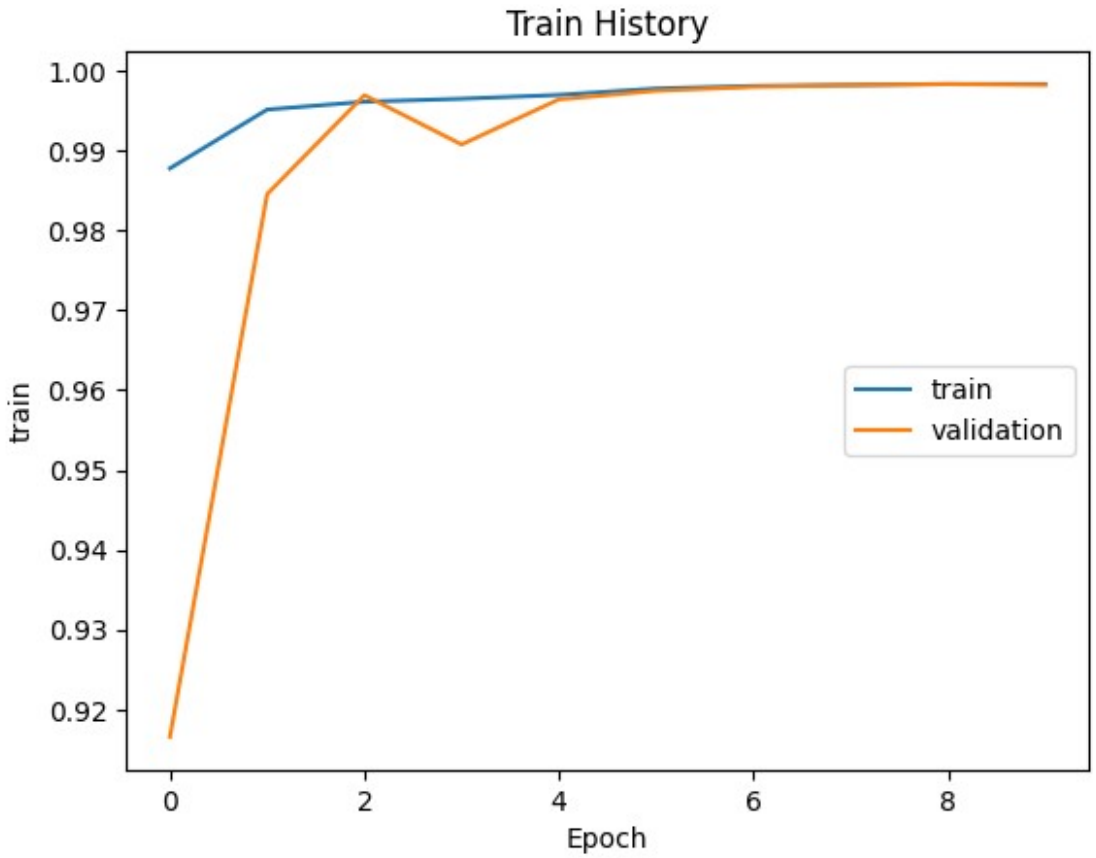

```
895
1/1 [==============================] - 0s 23ms/step
```



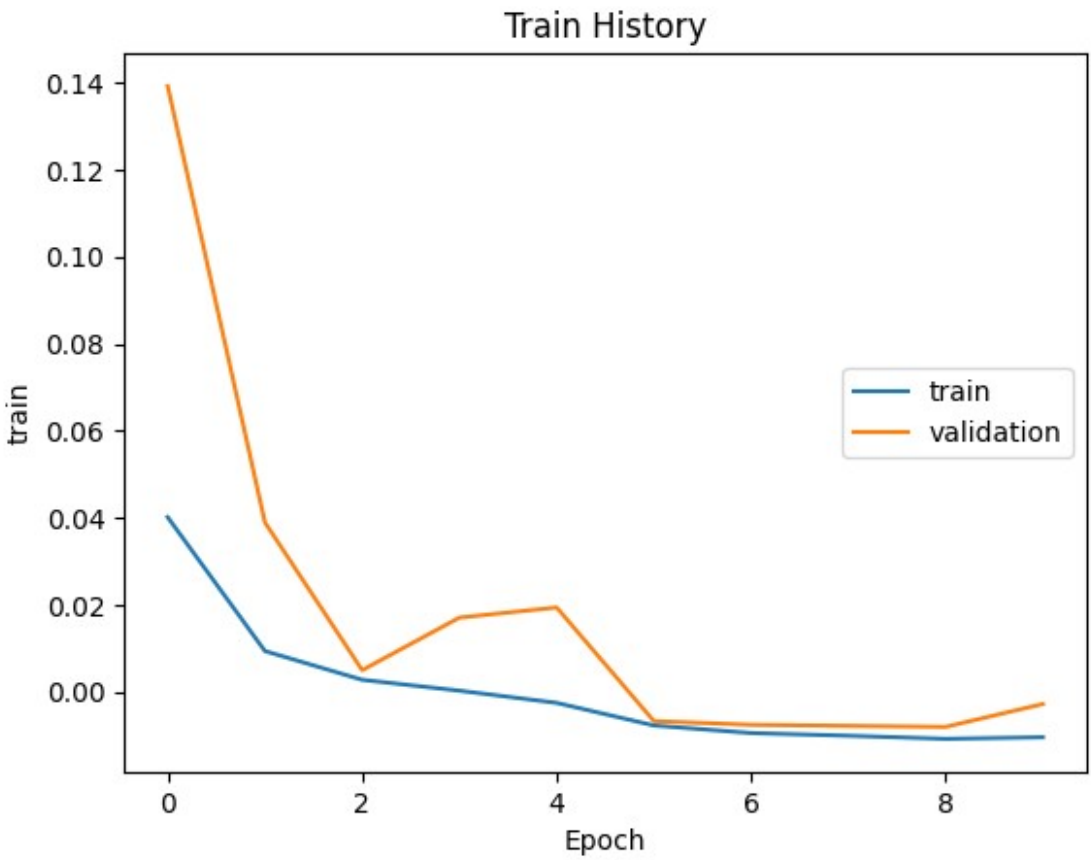

```
111
1/1 [==============================] - 0s 23ms/step
```




```
show_history(history7, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history7, 'loss', 'val_loss')
```




```
import tensorflow as tf
from tensorflow import keras

def conv_block(x, filters, kernel_size=(3, 3), padding='same',
strides=1):
    x = keras.layers.Conv2D(filters, kernel_size, padding=padding,
strides=strides)(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Activation('relu')(x)
    return x

def upconv_block(x, filters, kernel_size=(2, 2), padding='same'):
    x = keras.layers.Conv2DTranspose(filters, kernel_size, strides=(2,
2), padding=padding)(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Activation('relu')(x)
    return x

def vnet():
    inputs = keras.layers.Input((128, 128, 3))

    # Downward path
    conv1 = conv_block(inputs, 16)
```

```
    down1 = keras.layers.MaxPooling2D((2, 2))(conv1)

    conv2 = conv_block(down1, 32)
    down2 = keras.layers.MaxPooling2D((2, 2))(conv2)

    conv3 = conv_block(down2, 64)
    down3 = keras.layers.MaxPooling2D((2, 2))(conv3)

    conv4 = conv_block(down3, 128)
    down4 = keras.layers.MaxPooling2D((2, 2))(conv4)

    # Upward path
    up4 = upconv_block(down4, 128)
    concat4 = keras.layers.concatenate([conv4, up4], axis=-1)
    upconv4 = conv_block(concat4, 128)

    up3 = upconv_block(upconv4, 64)
    concat3 = keras.layers.concatenate([conv3, up3], axis=-1)
    upconv3 = conv_block(concat3, 64)

    up2 = upconv_block(upconv3, 32)
    concat2 = keras.layers.concatenate([conv2, up2], axis=-1)
    upconv2 = conv_block(concat2, 32)

    up1 = upconv_block(upconv2, 16)
    concat1 = keras.layers.concatenate([conv1, up1], axis=-1)
    upconv1 = conv_block(concat1, 16)

    outputs = keras.layers.Conv2D(1, (1, 1), activation='sigmoid')
(upconv1)

    model = keras.models.Model(inputs, outputs)
    return model

# Instantiate the model
model8 = vnet()

# Compile the model
model8.compile(optimizer='Adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Display the model summary
model8.summary()
```

```
Model: "model_18"
__________________________________________________________________________
_____________________________
 Layer (type)                Output Shape                  Param #
Connected to
==========================================================================
```

```
==============================
 input_19 (InputLayer)       [(None, 128, 128, 3)]        0          []


 conv2d_725 (Conv2D)         (None, 128, 128, 16)         448
['input_19[0][0]']


 batch_normalization_773 (B  (None, 128, 128, 16)         64
['conv2d_725[0][0]']
 atchNormalization)


 activation_718 (Activation  (None, 128, 128, 16)         0
['batch_normalization_773[0][0
 )                                                                ]'
]


 max_pooling2d_23 (MaxPooli  (None, 64, 64, 16)           0
['activation_718[0][0]']
 ng2D)


 conv2d_726 (Conv2D)         (None, 64, 64, 32)           4640
['max_pooling2d_23[0][0]']


 batch_normalization_774 (B  (None, 64, 64, 32)           128
['conv2d_726[0][0]']
 atchNormalization)


 activation_719 (Activation  (None, 64, 64, 32)           0
['batch_normalization_774[0][0
 )                                                                ]'
]


 max_pooling2d_24 (MaxPooli  (None, 32, 32, 32)           0
['activation_719[0][0]']
 ng2D)


 conv2d_727 (Conv2D)         (None, 32, 32, 64)           18496
['max_pooling2d_24[0][0]']
```

```
 batch_normalization_775 (B  (None, 32, 32, 64)           256
['conv2d_727[0][0]']
 atchNormalization)


 activation_720 (Activation  (None, 32, 32, 64)           0
['batch_normalization_775[0][0
 )                                                                ]'
]


 max_pooling2d_25 (MaxPooli  (None, 16, 16, 64)           0
['activation_720[0][0]']
 ng2D)


 conv2d_728 (Conv2D)         (None, 16, 16, 128)          73856
['max_pooling2d_25[0][0]']


 batch_normalization_776 (B  (None, 16, 16, 128)          512
['conv2d_728[0][0]']
 atchNormalization)


 activation_721 (Activation  (None, 16, 16, 128)          0
['batch_normalization_776[0][0
 )                                                                ]'
]


 max_pooling2d_26 (MaxPooli  (None, 8, 8, 128)            0
['activation_721[0][0]']
 ng2D)


 conv2d_transpose (Conv2DTr  (None, 16, 16, 128)          65664
['max_pooling2d_26[0][0]']
 anspose)


 batch_normalization_777 (B  (None, 16, 16, 128)          512
['conv2d_transpose[0][0]']
 atchNormalization)
```

```
 activation_722 (Activation  (None, 16, 16, 128)          0
['batch_normalization_777[0][0
 )                                                                ]'
]


 concatenate_539 (Concatena  (None, 16, 16, 256)          0
['activation_721[0][0]',
 te)
'activation_722[0][0]']


 conv2d_729 (Conv2D)         (None, 16, 16, 128)          295040
['concatenate_539[0][0]']


 batch_normalization_778 (B  (None, 16, 16, 128)          512
['conv2d_729[0][0]']
 atchNormalization)



 activation_723 (Activation  (None, 16, 16, 128)          0
['batch_normalization_778[0][0
 )                                                                ]'
]


 conv2d_transpose_1 (Conv2D  (None, 32, 32, 64)           32832
['activation_723[0][0]']
 Transpose)



 batch_normalization_779 (B  (None, 32, 32, 64)           256
['conv2d_transpose_1[0][0]']
 atchNormalization)



 activation_724 (Activation  (None, 32, 32, 64)           0
['batch_normalization_779[0][0
 )                                                                ]'
]


 concatenate_540 (Concatena  (None, 32, 32, 128)          0
```

```
['activation_720[0][0]',
 te)
'activation_724[0][0]']


 conv2d_730 (Conv2D)         (None, 32, 32, 64)           73792
['concatenate_540[0][0]']


 batch_normalization_780 (B  (None, 32, 32, 64)           256
['conv2d_730[0][0]']
 atchNormalization)



 activation_725 (Activation  (None, 32, 32, 64)           0
['batch_normalization_780[0][0
 )                                                                ]'
]


 conv2d_transpose_2 (Conv2D  (None, 64, 64, 32)           8224
['activation_725[0][0]']
 Transpose)



 batch_normalization_781 (B  (None, 64, 64, 32)           128
['conv2d_transpose_2[0][0]']
 atchNormalization)



 activation_726 (Activation  (None, 64, 64, 32)           0
['batch_normalization_781[0][0
 )                                                                ]'
]


 concatenate_541 (Concatena  (None, 64, 64, 64)           0
['activation_719[0][0]',
 te)
'activation_726[0][0]']


 conv2d_731 (Conv2D)         (None, 64, 64, 32)           18464
['concatenate_541[0][0]']


 batch_normalization_782 (B  (None, 64, 64, 32)           128
```

```
['conv2d_731[0][0]']
 atchNormalization)



 activation_727 (Activation  (None, 64, 64, 32)           0
['batch_normalization_782[0][0
 )                                                                ]'
]


 conv2d_transpose_3 (Conv2D  (None, 128, 128, 16)         2064
['activation_727[0][0]']
 Transpose)



 batch_normalization_783 (B  (None, 128, 128, 16)         64
['conv2d_transpose_3[0][0]']
 atchNormalization)



 activation_728 (Activation  (None, 128, 128, 16)         0
['batch_normalization_783[0][0
 )                                                                ]'
]


 concatenate_542 (Concatena  (None, 128, 128, 32)         0
['activation_718[0][0]',
 te)
'activation_728[0][0]']


 conv2d_732 (Conv2D)         (None, 128, 128, 16)         4624
['concatenate_542[0][0]']


 batch_normalization_784 (B  (None, 128, 128, 16)         64
['conv2d_732[0][0]']
 atchNormalization)



 activation_729 (Activation  (None, 128, 128, 16)         0
['batch_normalization_784[0][0
 )                                                                ]'
]
```

```
 conv2d_733 (Conv2D)         (None, 128, 128, 1)          17
['activation_729[0][0]']


==================================================================
============================
Total params: 601041 (2.29 MB)
Trainable params: 599601 (2.29 MB)
Non-trainable params: 1440 (5.62 KB)
__________________________________________________________________
_____________________________
```

```
model8.compile(optimizer=Adam(lr=0.001, beta_1=0.9,
beta_2=0.999),loss= 'binary_crossentropy', metrics=[dice_coefficient])
model8.save('Vnet_model8.h5')
```

```
checkpoint = tf.keras.callbacks.ModelCheckpoint("Vnet_model8.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                  patience=5,
                                                  mode='auto',

restore_best_weights=True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                 factor=0.2,
                                                 patience=2,
                                                 min_lr=1e-6,
                                                 mode='auto',
                                                 verbose=1)
# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]
```

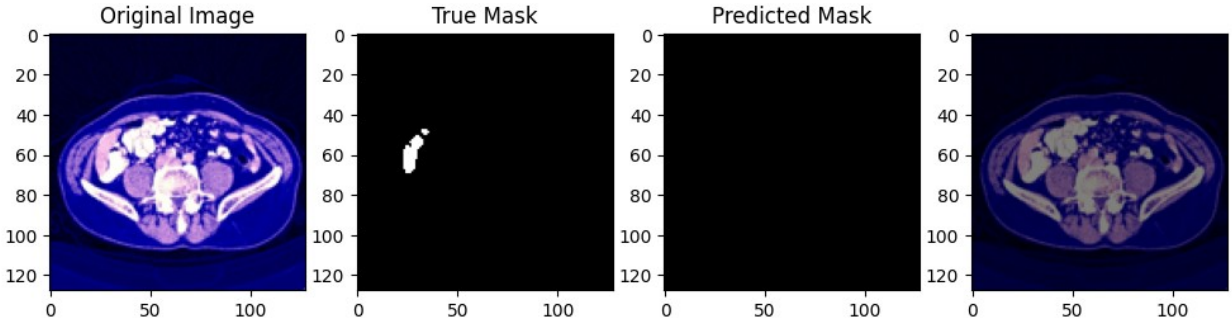
```
history8= model8.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model8.save('Vnet_model8.h5')
```

```
Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.2086 -
dice_coefficient: 0.8637
Epoch 1: val_loss improved from inf to 0.19796, saving model to
Vnet_model8.h5
394/394 [==============================] - 20s 27ms/step - loss:
0.2086 - dice_coefficient: 0.8637 - val_loss: 0.1980 -
val_dice_coefficient: 0.9167 - lr: 0.0010
Epoch 2/10
394/394 [==============================] - ETA: 0s - loss: 0.0405 -
```

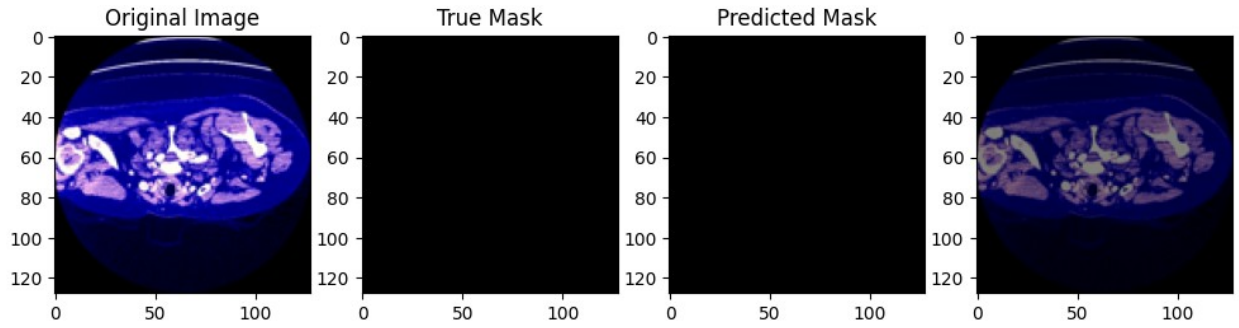
```
dice_coefficient: 0.9666
Epoch 2: val_loss improved from 0.19796 to 0.02173, saving model to
Vnet_model8.h5
394/394 [==============================] - 8s 21ms/step - loss: 0.0405
- dice_coefficient: 0.9666 - val_loss: 0.0217 - val_dice_coefficient:
0.9772 - lr: 0.0010
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: 0.0078 -
dice_coefficient: 0.9854
Epoch 3: val_loss improved from 0.02173 to -0.00926, saving model to
Vnet_model8.h5
394/394 [==============================] - 8s 21ms/step - loss: 0.0078
- dice_coefficient: 0.9854 - val_loss: -0.0093 - val_dice_coefficient:
0.9899 - lr: 0.0010
Epoch 4/10
394/394 [==============================] - ETA: 0s - loss: -0.0195 -
dice_coefficient: 0.9902
Epoch 4: val_loss improved from -0.00926 to -0.11057, saving model to
Vnet_model8.h5
394/394 [==============================] - 8s 21ms/step - loss: -
0.0195 - dice_coefficient: 0.9902 - val_loss: -0.1106 -
val_dice_coefficient: 0.9891 - lr: 0.0010
Epoch 5/10
394/394 [==============================] - ETA: 0s - loss: -0.0562 -
dice_coefficient: 0.9916
Epoch 5: val_loss improved from -0.11057 to -0.14068, saving model to
Vnet_model8.h5
394/394 [==============================] - 8s 21ms/step - loss: -
0.0562 - dice_coefficient: 0.9916 - val_loss: -0.1407 -
val_dice_coefficient: 0.9924 - lr: 0.0010
Epoch 6/10
394/394 [==============================] - ETA: 0s - loss: -0.1331 -
dice_coefficient: 0.9932
Epoch 6: val_loss did not improve from -0.14068
394/394 [==============================] - 8s 21ms/step - loss: -
0.1331 - dice_coefficient: 0.9932 - val_loss: 0.0085 -
val_dice_coefficient: 0.9901 - lr: 0.0010
Epoch 7/10
394/394 [==============================] - ETA: 0s - loss: -0.2399 -
dice_coefficient: 0.9930
Epoch 7: val_loss improved from -0.14068 to -0.30531, saving model to
Vnet_model8.h5
394/394 [==============================] - 8s 21ms/step - loss: -
0.2399 - dice_coefficient: 0.9930 - val_loss: -0.3053 -
val_dice_coefficient: 0.9946 - lr: 0.0010
Epoch 8/10
394/394 [==============================] - ETA: 0s - loss: -0.3041 -
dice_coefficient: 0.9922
Epoch 8: val_loss improved from -0.30531 to -0.86572, saving model to
```

```
Vnet_model8.h5
394/394 [==============================] - 8s 21ms/step - loss: -
0.3041 - dice_coefficient: 0.9922 - val_loss: -0.8657 -
val_dice_coefficient: 0.9888 - lr: 0.0010
Epoch 9/10
394/394 [==============================] - ETA: 0s - loss: -0.5192 -
dice_coefficient: 0.9931
Epoch 9: val_loss did not improve from -0.86572
394/394 [==============================] - 8s 21ms/step - loss: -
0.5192 - dice_coefficient: 0.9931 - val_loss: 1.0598 -
val_dice_coefficient: 0.9900 - lr: 0.0010
Epoch 10/10
394/394 [==============================] - ETA: 0s - loss: -0.7264 -
dice_coefficient: 0.9939
Epoch 10: val_loss improved from -0.86572 to -1.24591, saving model to
Vnet_model8.h5
394/394 [==============================] - 8s 21ms/step - loss: -
0.7264 - dice_coefficient: 0.9939 - val_loss: -1.2459 -
val_dice_coefficient: 0.9943 - lr: 0.0010

scores8 = model8.evaluate(x_valid, y_valid)
scores8[1]

57/57 [==============================] - 1s 12ms/step - loss: -1.2459
- dice_coefficient: 0.9943

0.9942881464958191

prediction8 = model8.predict(x_test)

test_scores8 = model8.evaluate(x_test, y_test)
test_scores8[1]

29/29 [==============================] - 1s 17ms/step
29/29 [==============================] - 0s 12ms/step - loss: -1.2074
- dice_coefficient: 0.9940

0.9939619302749634

import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 20, replace=False)
for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
```

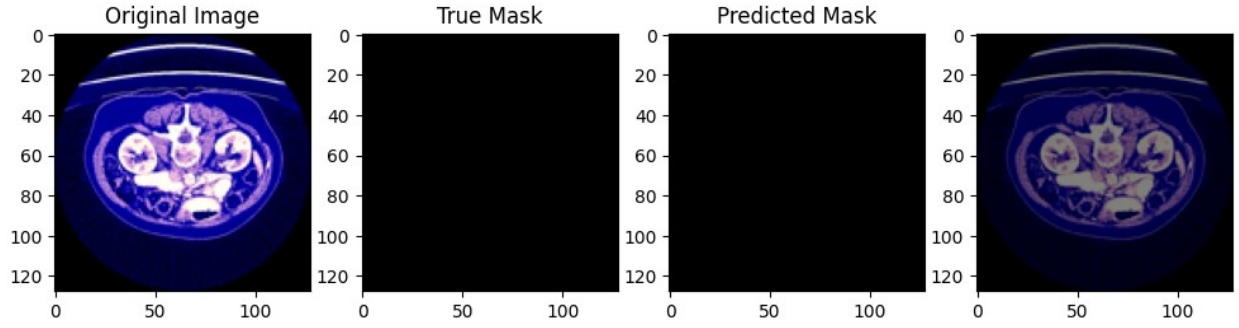
```
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model3
    predicted_mask = model8.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
    threshold = 0.5  # Adjust this threshold based on your model's
output
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')
    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
'nipy_spectral')
    plt.show()

333
1/1 [==============================] - 0s 177ms/step
```



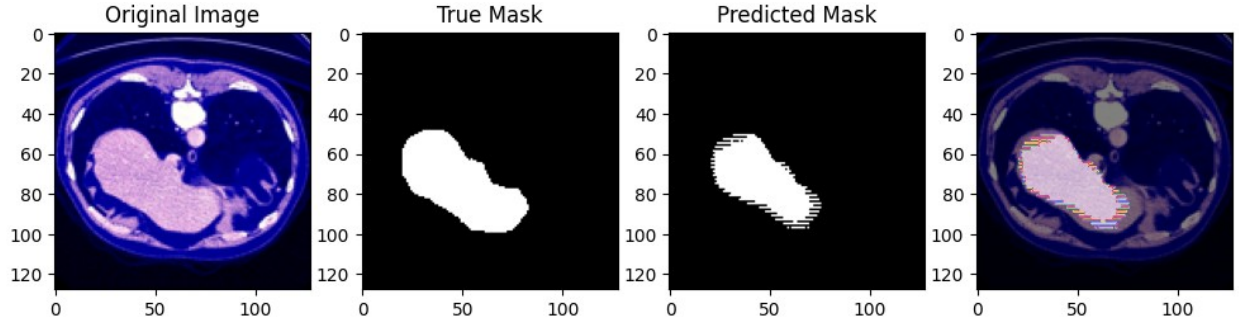

```
812
1/1 [==============================] - 0s 25ms/step
```



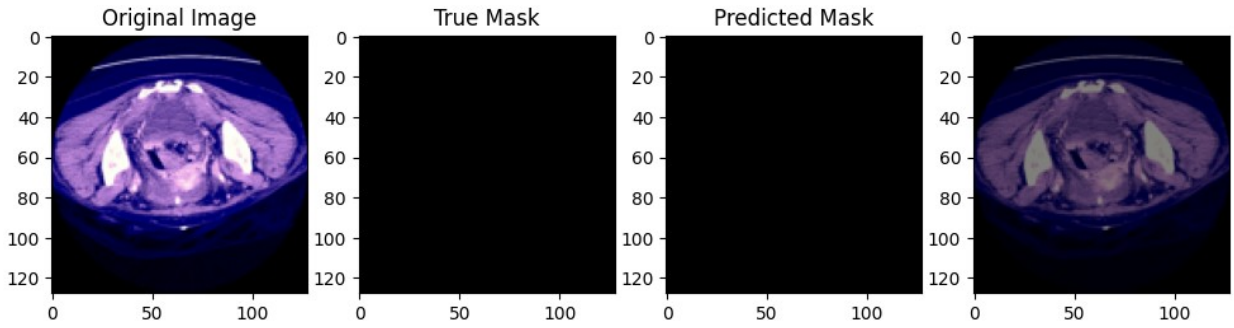

```
745
1/1 [==============================] - 0s 22ms/step
```



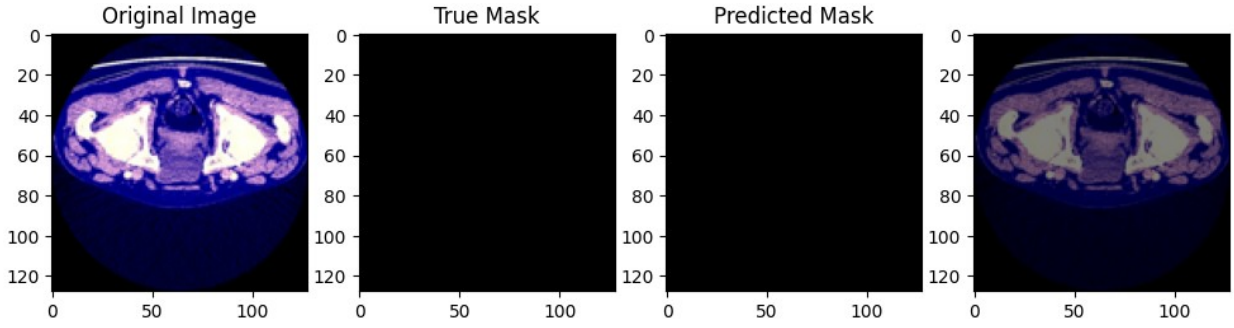

```
245
1/1 [==============================] - 0s 25ms/step
```



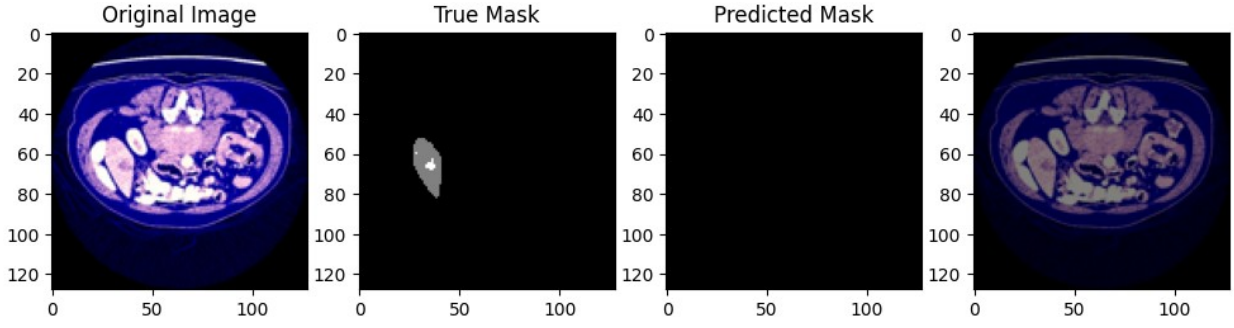

```
40
1/1 [==============================] - 0s 24ms/step
```



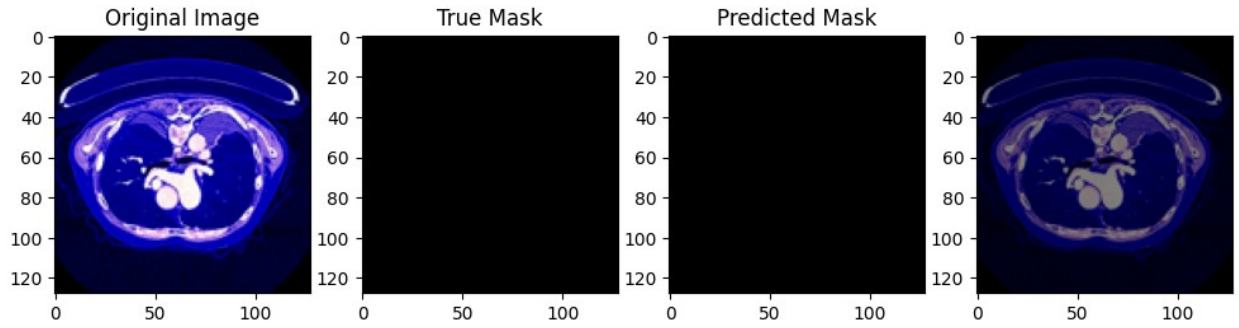

```
427
1/1 [==============================] - 0s 27ms/step
```



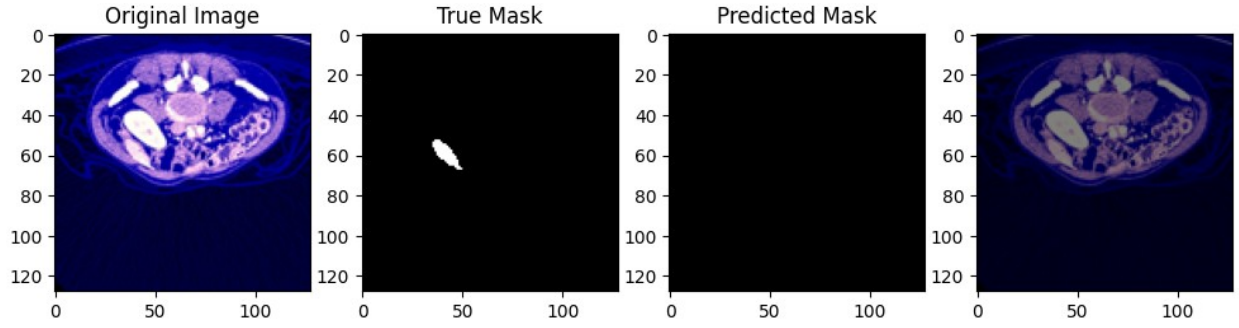

```
303
1/1 [==============================] - 0s 27ms/step
```



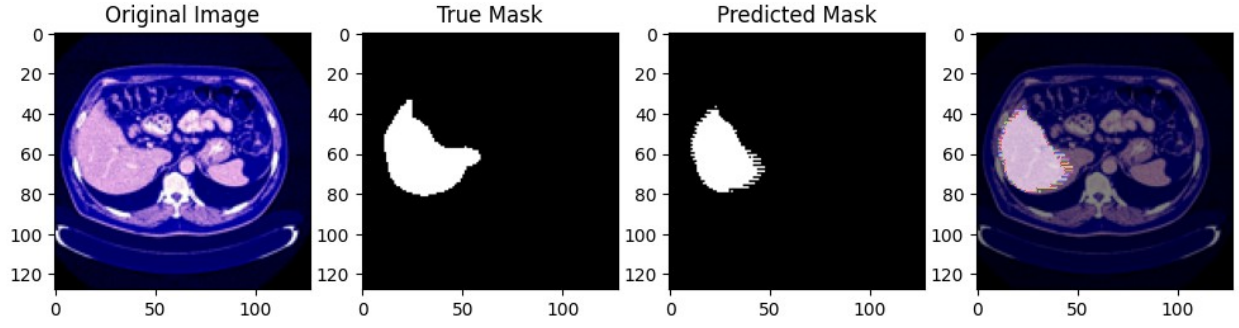

```
588
1/1 [==============================] - 0s 31ms/step
```



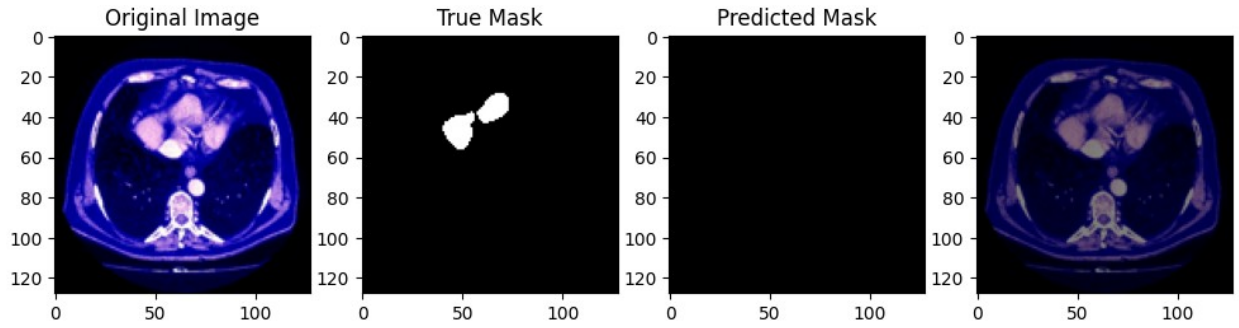

```
210
1/1 [==============================] - 0s 25ms/step
```



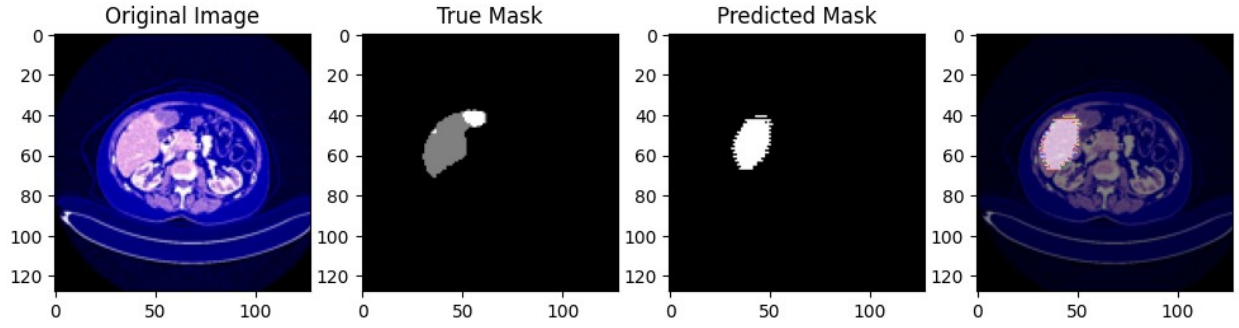

```
137
1/1 [==============================] - 0s 30ms/step
```



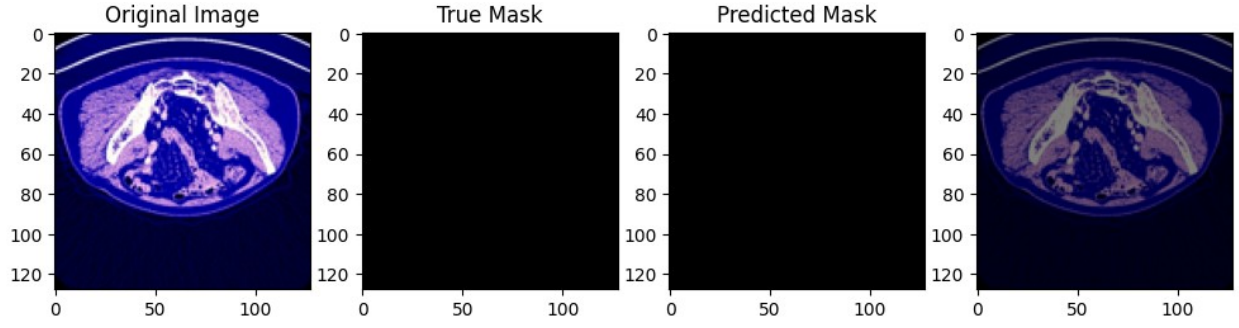

```
138
1/1 [==============================] - 0s 25ms/step
```



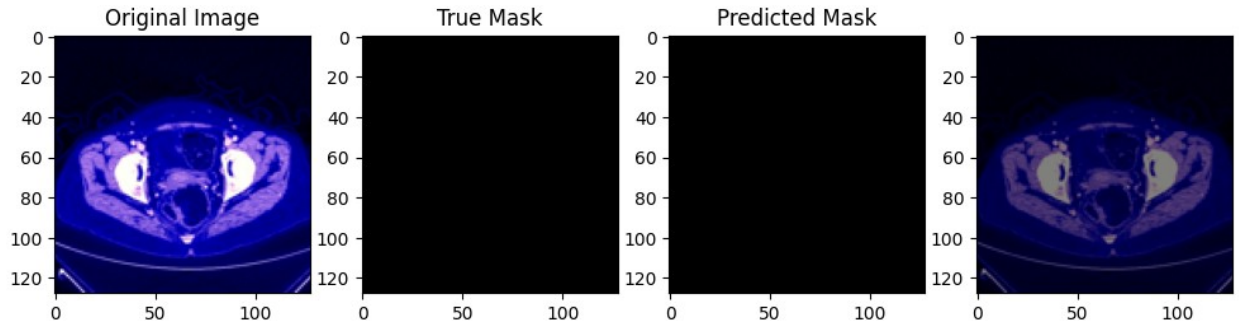

```
261
1/1 [==============================] - 0s 26ms/step
```



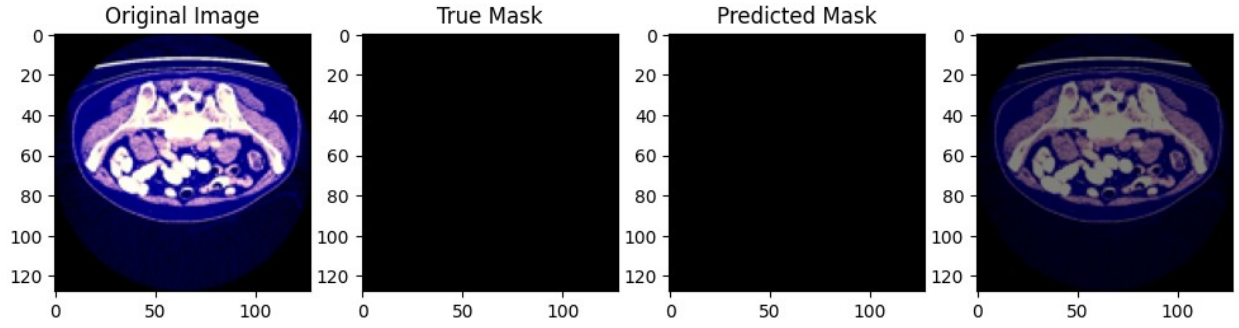

```
352
1/1 [==============================] - 0s 24ms/step
```



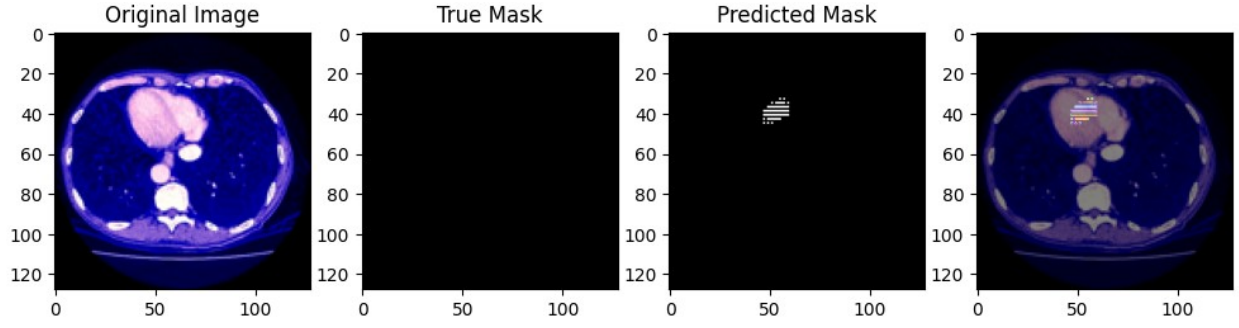

```
360
1/1 [==============================] - 0s 24ms/step
```



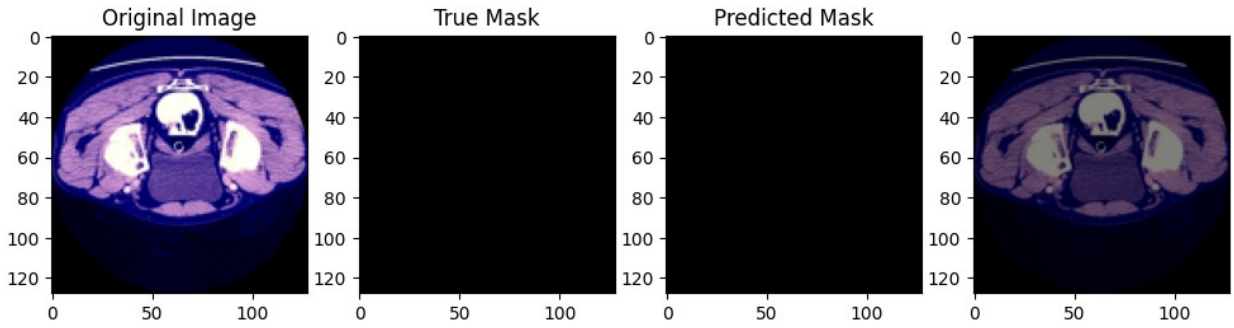

```
347
1/1 [==============================] - 0s 24ms/step
```



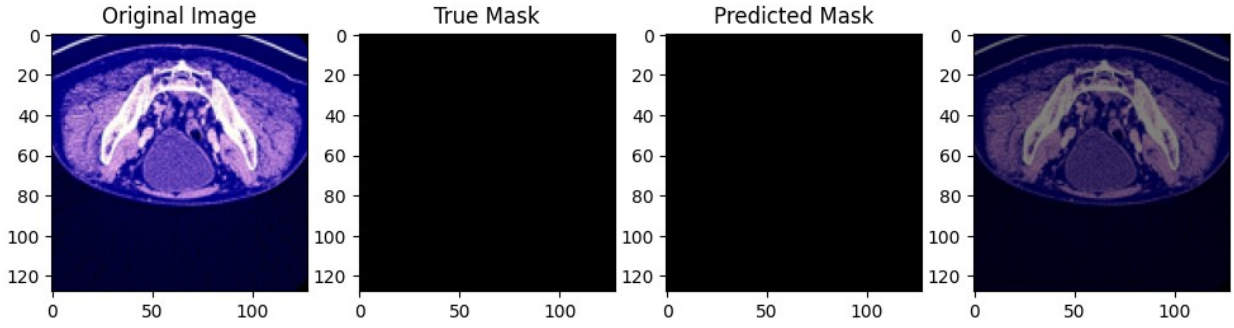

```
479
1/1 [==============================] - 0s 23ms/step
```



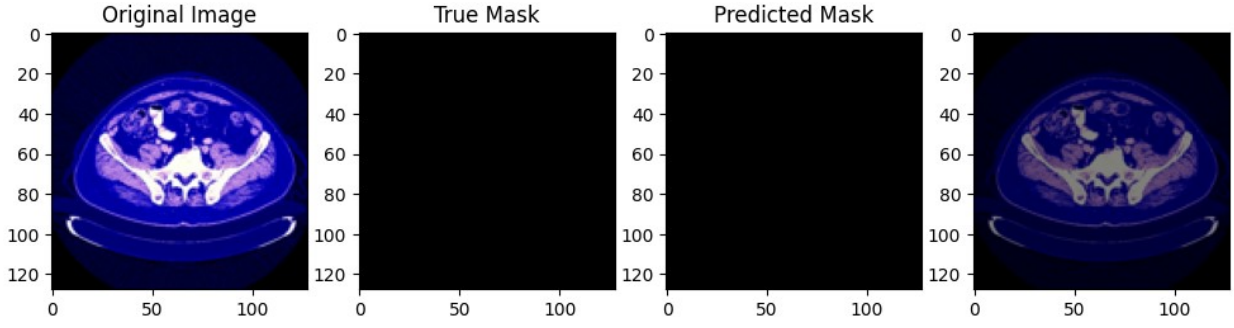

```
307
1/1 [==============================] - 0s 23ms/step
```



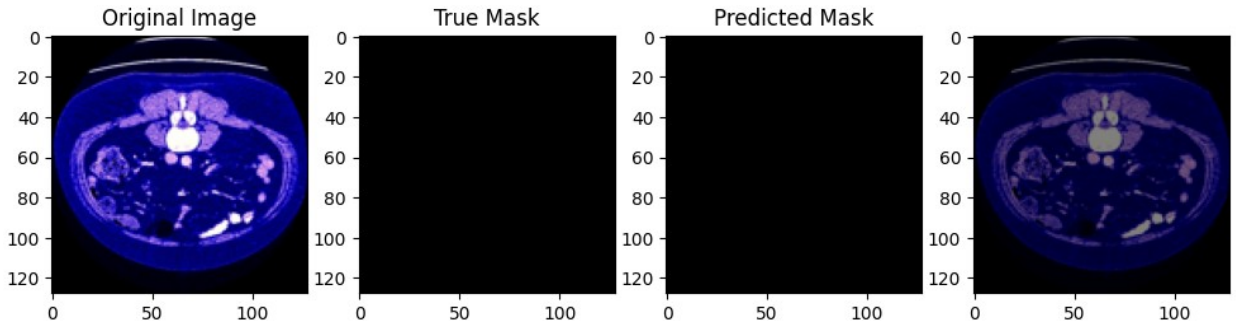

```
658
1/1 [==============================] - 0s 22ms/step
```



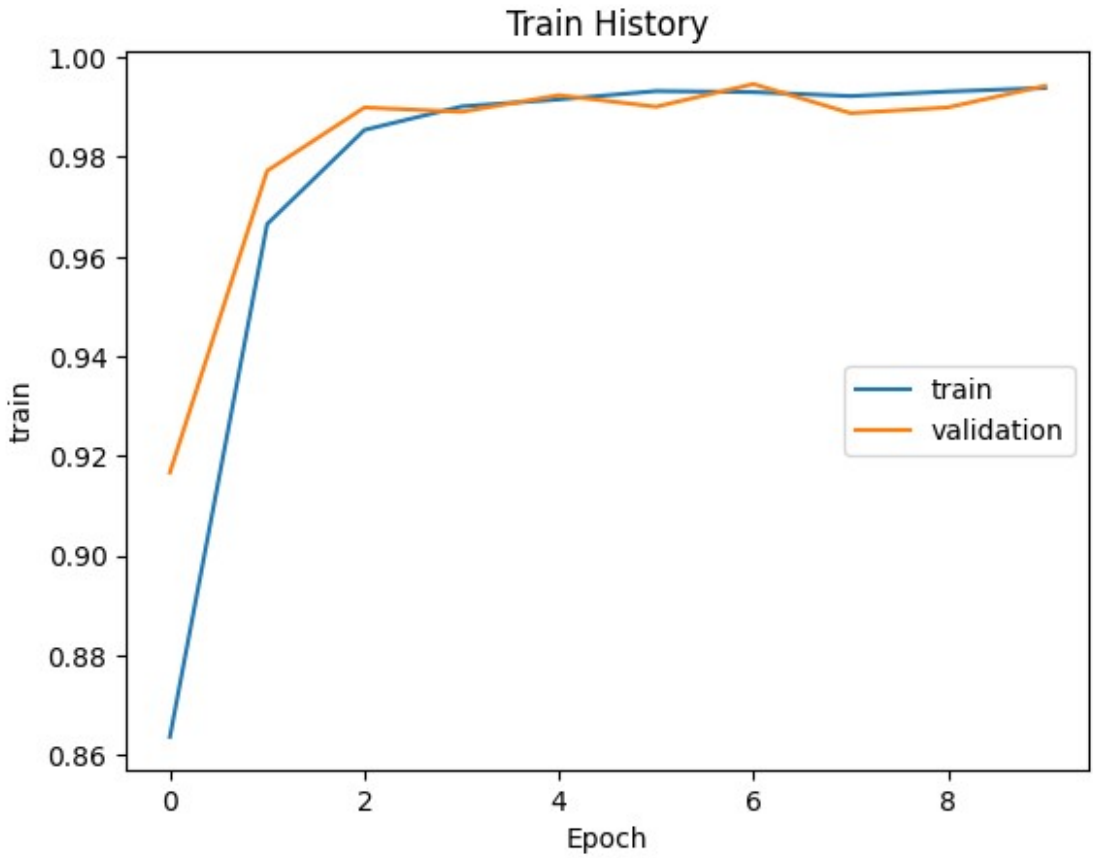

```
895
1/1 [==============================] - 0s 23ms/step
```



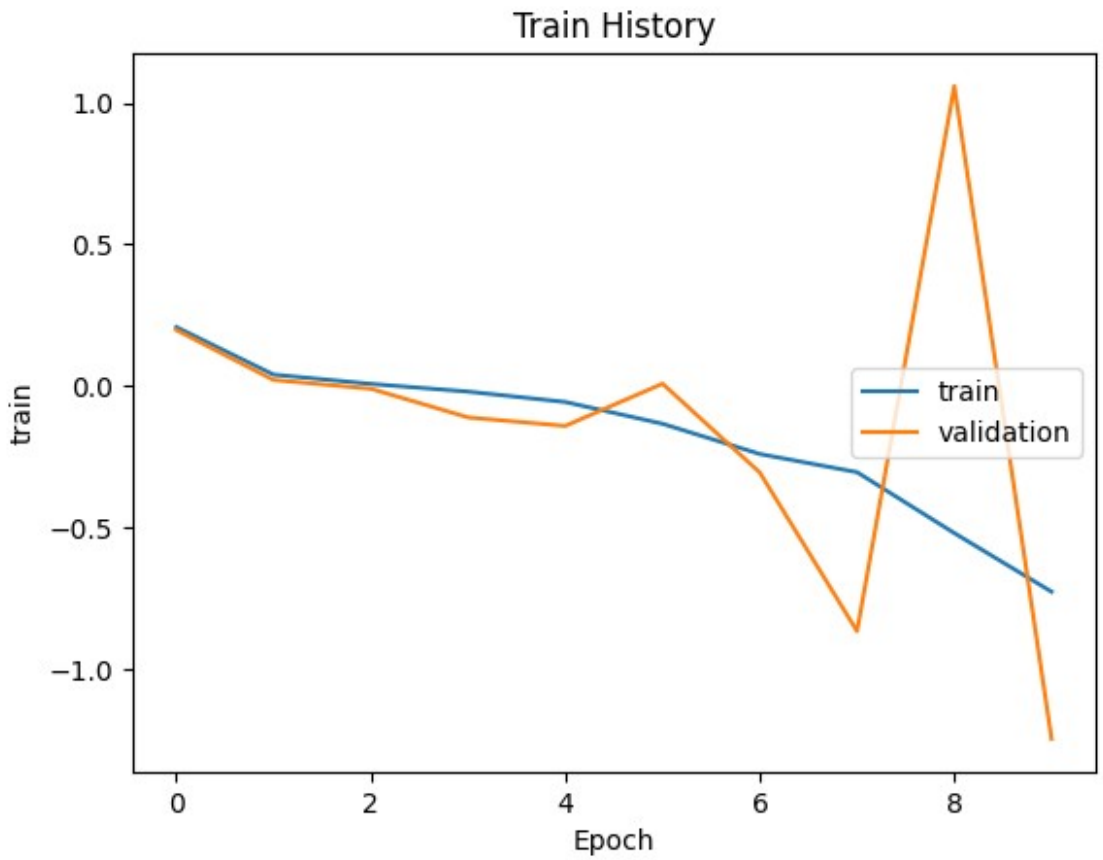

```
111
1/1 [==============================] - 0s 27ms/step
```




```
show_history(history8, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history8, 'loss', 'val_loss')
```




```
import tensorflow as tf
from tensorflow import keras

def conv_block(inputs, filters, kernel_size=(3, 3), strides=(1, 1),
padding='same'):
    x = keras.layers.Conv2D(filters, kernel_size, strides=strides,
padding=padding)(inputs)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.ReLU()(x)
    return x

def stage(inputs, filters, num_blocks, stage_name):
    x = inputs
    for i in range(num_blocks):
        x = conv_block(x, filters, kernel_size=(3, 3), padding='same',
strides=(1, 1))
    return x

def transition_down(inputs, filters, stage_name):
    x = conv_block(inputs, filters, kernel_size=(3, 3),
padding='same', strides=(2, 2))
    return x
```

```
def transition_up(inputs, filters, stage_name, scale_factor):
    x = keras.layers.Conv2DTranspose(filters, (3, 3),
strides=(scale_factor, scale_factor), padding='same')(inputs)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.ReLU()(x)
    return x

def HRNet(input_shape=(128, 128, 3)):
    inputs = keras.layers.Input(shape=input_shape)

    # Stage 1
    x = conv_block(inputs, 64, kernel_size=(3, 3), padding='same',
strides=(2, 2))
    x = conv_block(x, 64, kernel_size=(3, 3), padding='same',
strides=(2, 2))

    # Stage 2
    stage2_out = stage(x, 32, num_blocks=4, stage_name='stage2')
    stage2_out_td = transition_down(stage2_out, 64,
stage_name='stage2')

    # Stage 3
    stage3_out = stage(stage2_out_td, 32, num_blocks=4,
stage_name='stage3')
    stage3_out_td = transition_down(stage3_out, 128,
stage_name='stage3')

    # Stage 4
    stage4_out = stage(stage3_out_td, 32, num_blocks=4,
stage_name='stage4')

    # Upsampling
    stage3_out_tu = transition_up(stage4_out, 64,
stage_name='stage3_transition_up', scale_factor=4)
    stage2_out_tu = transition_up(stage3_out_tu, 32,
stage_name='stage2_transition_up', scale_factor=4)

    # Final output
    outputs = keras.layers.Conv2D(1, (1, 1), activation='sigmoid',
name='output')(stage2_out_tu)

    model = keras.models.Model(inputs, outputs)
    return model


# Instantiate the model
model9 = HRNet()

# Compile the model
model9.compile(optimizer='Adam', loss='binary_crossentropy',
```

```
metrics=['accuracy'])

# Display the model summary
model9.summary()
```

```
Model: "model_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_22 (InputLayer)       [(None, 128, 128, 3)]     0

 conv2d_750 (Conv2D)         (None, 64, 64, 64)        1792

 batch_normalization_803 (B  (None, 64, 64, 64)        256
 atchNormalization)

 re_lu_30 (ReLU)             (None, 64, 64, 64)        0

 conv2d_751 (Conv2D)         (None, 32, 32, 64)        36928

 batch_normalization_804 (B  (None, 32, 32, 64)        256
 atchNormalization)

 re_lu_31 (ReLU)             (None, 32, 32, 64)        0

 conv2d_752 (Conv2D)         (None, 32, 32, 32)        18464

 batch_normalization_805 (B  (None, 32, 32, 32)        128
 atchNormalization)

 re_lu_32 (ReLU)             (None, 32, 32, 32)        0

 conv2d_753 (Conv2D)         (None, 32, 32, 32)        9248

 batch_normalization_806 (B  (None, 32, 32, 32)        128
 atchNormalization)

 re_lu_33 (ReLU)             (None, 32, 32, 32)        0

 conv2d_754 (Conv2D)         (None, 32, 32, 32)        9248

 batch_normalization_807 (B  (None, 32, 32, 32)        128
 atchNormalization)

 re_lu_34 (ReLU)             (None, 32, 32, 32)        0

 conv2d_755 (Conv2D)         (None, 32, 32, 32)        9248

 batch_normalization_808 (B  (None, 32, 32, 32)        128
 atchNormalization)
```

```
re_lu_35 (ReLU)              (None, 32, 32, 32)        0

conv2d_756 (Conv2D)          (None, 16, 16, 64)        18496

batch_normalization_809 (B  (None, 16, 16, 64)        256
atchNormalization)

re_lu_36 (ReLU)              (None, 16, 16, 64)        0

conv2d_757 (Conv2D)          (None, 16, 16, 32)        18464

batch_normalization_810 (B  (None, 16, 16, 32)        128
atchNormalization)

re_lu_37 (ReLU)              (None, 16, 16, 32)        0

conv2d_758 (Conv2D)          (None, 16, 16, 32)        9248

batch_normalization_811 (B  (None, 16, 16, 32)        128
atchNormalization)

re_lu_38 (ReLU)              (None, 16, 16, 32)        0

conv2d_759 (Conv2D)          (None, 16, 16, 32)        9248

batch_normalization_812 (B  (None, 16, 16, 32)        128
atchNormalization)

re_lu_39 (ReLU)              (None, 16, 16, 32)        0

conv2d_760 (Conv2D)          (None, 16, 16, 32)        9248

batch_normalization_813 (B  (None, 16, 16, 32)        128
atchNormalization)

re_lu_40 (ReLU)              (None, 16, 16, 32)        0

conv2d_761 (Conv2D)          (None, 8, 8, 128)         36992

batch_normalization_814 (B  (None, 8, 8, 128)         512
atchNormalization)

re_lu_41 (ReLU)              (None, 8, 8, 128)         0

conv2d_762 (Conv2D)          (None, 8, 8, 32)          36896

batch_normalization_815 (B  (None, 8, 8, 32)          128
atchNormalization)
```

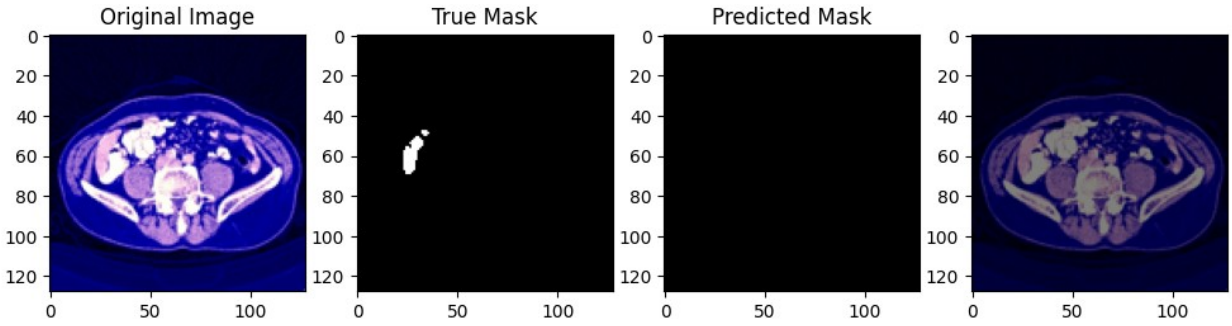
```
 re_lu_42 (ReLU)              (None, 8, 8, 32)          0

 conv2d_763 (Conv2D)          (None, 8, 8, 32)          9248

 batch_normalization_816 (B  (None, 8, 8, 32)          128
 atchNormalization)

 re_lu_43 (ReLU)              (None, 8, 8, 32)          0

 conv2d_764 (Conv2D)          (None, 8, 8, 32)          9248

 batch_normalization_817 (B  (None, 8, 8, 32)          128
 atchNormalization)

 re_lu_44 (ReLU)              (None, 8, 8, 32)          0

 conv2d_765 (Conv2D)          (None, 8, 8, 32)          9248

 batch_normalization_818 (B  (None, 8, 8, 32)          128
 atchNormalization)

 re_lu_45 (ReLU)              (None, 8, 8, 32)          0

 conv2d_transpose_6 (Conv2D  (None, 32, 32, 64)        18496
 Transpose)

 batch_normalization_819 (B  (None, 32, 32, 64)        256
 atchNormalization)

 re_lu_46 (ReLU)              (None, 32, 32, 64)        0

 conv2d_transpose_7 (Conv2D  (None, 128, 128, 32)      18464
 Transpose)

 batch_normalization_820 (B  (None, 128, 128, 32)      128
 atchNormalization)

 re_lu_47 (ReLU)              (None, 128, 128, 32)      0

 output (Conv2D)              (None, 128, 128, 1)       33

=================================================================
Total params: 291457 (1.11 MB)
Trainable params: 289857 (1.11 MB)
Non-trainable params: 1600 (6.25 KB)
_________________________________________________________________
```

```
model9.compile(optimizer=Adam(lr=0.001, beta_1=0.9,
beta_2=0.999),loss= 'binary_crossentropy', metrics=[dice_coefficient])
model9.save('HRnet_model9.h5')

checkpoint = tf.keras.callbacks.ModelCheckpoint("HRnet_model9.h5",
monitor='val_loss', verbose=1, patience = 3,save_best_only=True,
                                                mode='auto')

# Define other similar callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                   patience=5,
                                                   mode='auto',

restore_best_weights=True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                  factor=0.2,
                                                  patience=2,
                                                  min_lr=1e-6,
                                                  mode='auto',
                                                  verbose=1)
# Combine all callbacks into a list
callbacks = [checkpoint, early_stopping, reduce_lr]

history9= model9.fit(x_train, y_train, epochs=10, batch_size=16,
          validation_data=(x_valid, y_valid),
callbacks=[reduce_lr,checkpoint])
model9.save('HRnet_model9.h5')

Epoch 1/10
394/394 [==============================] - ETA: 0s - loss: 0.1409 -
dice_coefficient: 0.9254
Epoch 1: val_loss improved from inf to 0.08831, saving model to
HRnet_model9.h5
394/394 [==============================] - 23s 25ms/step - loss:
0.1409 - dice_coefficient: 0.9254 - val_loss: 0.0883 -
val_dice_coefficient: 0.9641 - lr: 0.0010
Epoch 2/10
392/394 [============================>.] - ETA: 0s - loss: 0.0861 -
dice_coefficient: 0.9713
Epoch 2: val_loss did not improve from 0.08831
394/394 [==============================] - 8s 20ms/step - loss: 0.0861
- dice_coefficient: 0.9713 - val_loss: 0.0959 - val_dice_coefficient:
0.9735 - lr: 0.0010
Epoch 3/10
394/394 [==============================] - ETA: 0s - loss: 0.0837 -
dice_coefficient: 0.9730
Epoch 3: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.

Epoch 3: val_loss did not improve from 0.08831
```

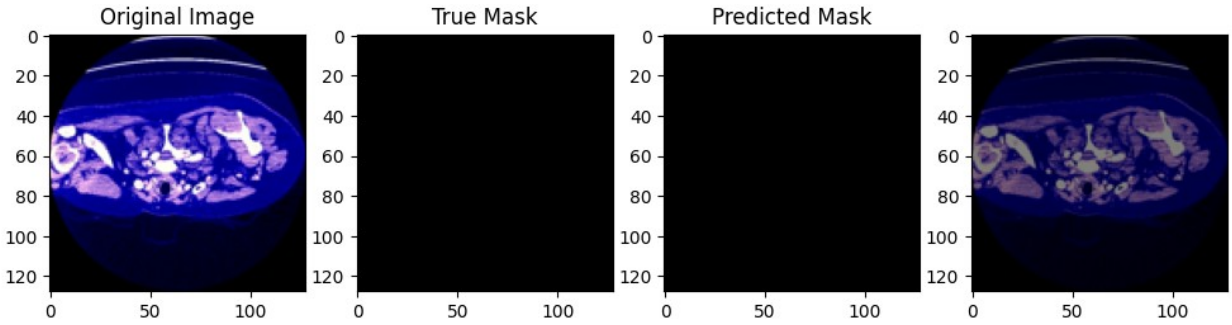
```
394/394 [==============================] - 8s 19ms/step - loss: 0.0837
- dice_coefficient: 0.9730 - val_loss: 0.0889 - val_dice_coefficient:
0.9743 - lr: 0.0010
Epoch 4/10
392/394 [============================>.] - ETA: 0s - loss: 0.0744 -
dice_coefficient: 0.9740
Epoch 4: val_loss improved from 0.08831 to 0.06034, saving model to
HRnet_model9.h5
394/394 [==============================] - 8s 20ms/step - loss: 0.0744
- dice_coefficient: 0.9740 - val_loss: 0.0603 - val_dice_coefficient:
0.9748 - lr: 2.0000e-04
Epoch 5/10
394/394 [==============================] - ETA: 0s - loss: 0.0671 -
dice_coefficient: 0.9736
Epoch 5: val_loss improved from 0.06034 to 0.04275, saving model to
HRnet_model9.h5
394/394 [==============================] - 8s 20ms/step - loss: 0.0671
- dice_coefficient: 0.9736 - val_loss: 0.0427 - val_dice_coefficient:
0.9744 - lr: 2.0000e-04
Epoch 6/10
393/394 [============================>.] - ETA: 0s - loss: 0.0585 -
dice_coefficient: 0.9736
Epoch 6: val_loss improved from 0.04275 to 0.03168, saving model to
HRnet_model9.h5
394/394 [==============================] - 8s 20ms/step - loss: 0.0584
- dice_coefficient: 0.9736 - val_loss: 0.0317 - val_dice_coefficient:
0.9751 - lr: 2.0000e-04
Epoch 7/10
393/394 [============================>.] - ETA: 0s - loss: 0.0496 -
dice_coefficient: 0.9740
Epoch 7: val_loss improved from 0.03168 to 0.01002, saving model to
HRnet_model9.h5
394/394 [==============================] - 8s 20ms/step - loss: 0.0496
- dice_coefficient: 0.9740 - val_loss: 0.0100 - val_dice_coefficient:
0.9743 - lr: 2.0000e-04
Epoch 8/10
394/394 [==============================] - ETA: 0s - loss: 0.0366 -
dice_coefficient: 0.9737
Epoch 8: val_loss improved from 0.01002 to -0.01812, saving model to
HRnet_model9.h5
394/394 [==============================] - 8s 20ms/step - loss: 0.0366
- dice_coefficient: 0.9737 - val_loss: -0.0181 - val_dice_coefficient:
0.9745 - lr: 2.0000e-04
Epoch 9/10
394/394 [==============================] - ETA: 0s - loss: 0.0252 -
dice_coefficient: 0.9729
Epoch 9: val_loss improved from -0.01812 to -0.01927, saving model to
HRnet_model9.h5
394/394 [==============================] - 8s 20ms/step - loss: 0.0252
```

```
- dice_coefficient: 0.9729 - val_loss: -0.0193 - val_dice_coefficient:
0.9761 - lr: 2.0000e-04
Epoch 10/10
391/394 [============================>.] - ETA: 0s - loss: 0.0125 -
dice_coefficient: 0.9743
Epoch 10: val_loss improved from -0.01927 to -0.03133, saving model to
HRnet_model9.h5
394/394 [==============================] - 8s 20ms/step - loss: 0.0126
- dice_coefficient: 0.9743 - val_loss: -0.0313 - val_dice_coefficient:
0.9751 - lr: 2.0000e-04

scores9 = model9.evaluate(x_valid, y_valid)
scores9[1]

57/57 [==============================] - 1s 10ms/step - loss: -0.0313
- dice_coefficient: 0.9751

0.9751148223876953

prediction9 = model9.predict(x_test)

test_scores9 = model9.evaluate(x_test, y_test)
test_scores9[1]

29/29 [==============================] - 1s 14ms/step
29/29 [==============================] - 0s 10ms/step - loss: -0.0203
- dice_coefficient: 0.9745

0.9745174646377563

import numpy as np
import random
import matplotlib.pyplot as plt
import random
image_list=random.sample(range(1, 900), 20)
import numpy as np
np.random.seed(42)
image_list = np.random.choice(range(1, 900), 20, replace=False)
for i in (image_list):
# Assuming you have x_test and y_test
    print(i)
    image_index = i
    # Load the image and true mask
    input_image = x_valid[image_index]
    true_mask = y_valid[image_index]
    # Obtain the predicted mask from model3
    predicted_mask = model9.predict(np.expand_dims(input_image,
axis=0))[0]
    # Threshold the predicted mask
    threshold = 0.5  # Adjust this threshold based on your model's
output
```

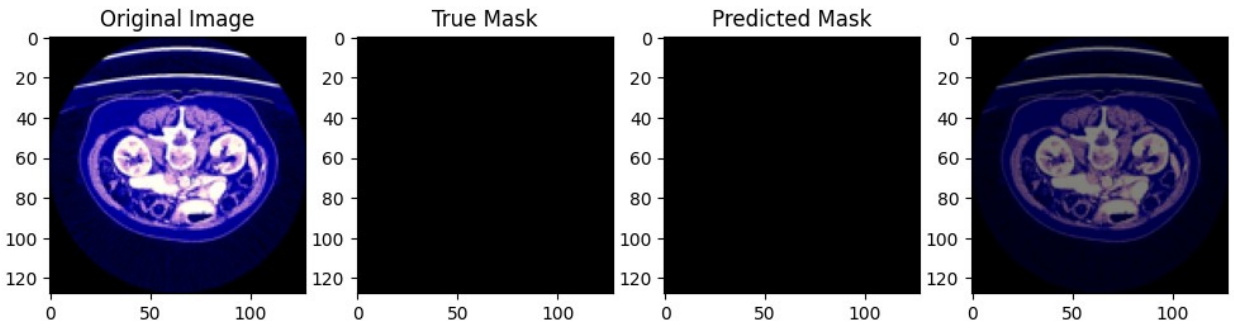
```
    predicted_mask_binary = (predicted_mask >
threshold).astype(np.uint8)
    # Plotting
    plt.figure(figsize=(12, 4))
    # original image
    plt.subplot(1, 4, 1)
    plt.imshow(input_image)
    plt.title('Original Image')
    # true mask
    plt.subplot(1, 4, 2)
    plt.imshow(true_mask[:, :, 0], cmap='gray')
    plt.title('True Mask')
    #predicted mask
    plt.subplot(1, 4, 3)
    plt.imshow(predicted_mask_binary[:, :, 0], cmap='gray')
    plt.title('Predicted Mask')
    plt.subplot(1,4,4)
    plt.imshow(input_image,cmap='bone')
    plt.imshow(predicted_mask_binary[:,:,0],alpha=0.5,cmap =
'nipy_spectral')
    plt.show()

333
1/1 [==============================] - 0s 141ms/step
```



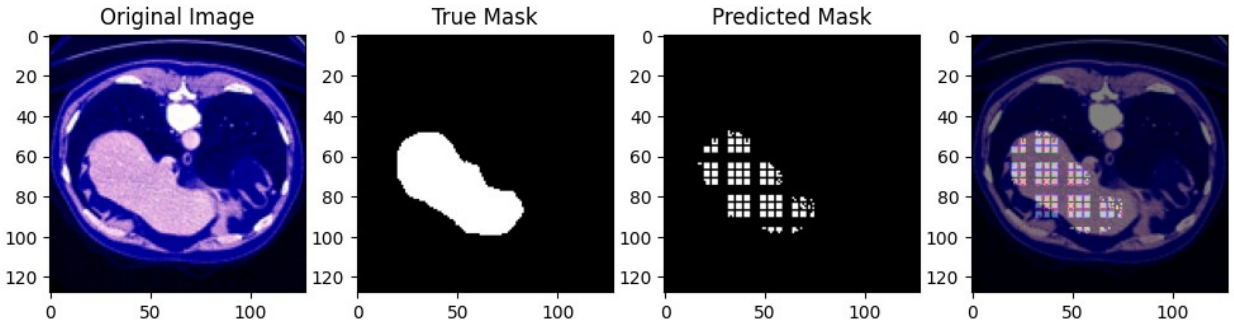

```
812
1/1 [==============================] - 0s 25ms/step
```



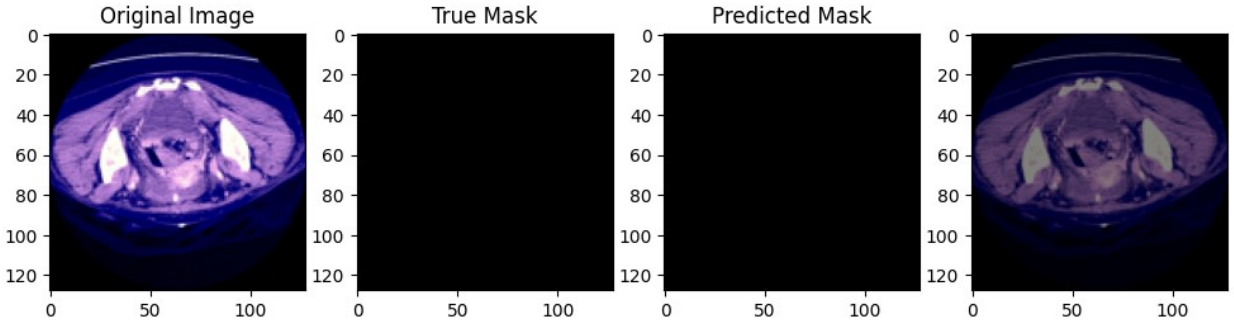
```
745
1/1 [==============================] - 0s 25ms/step
```



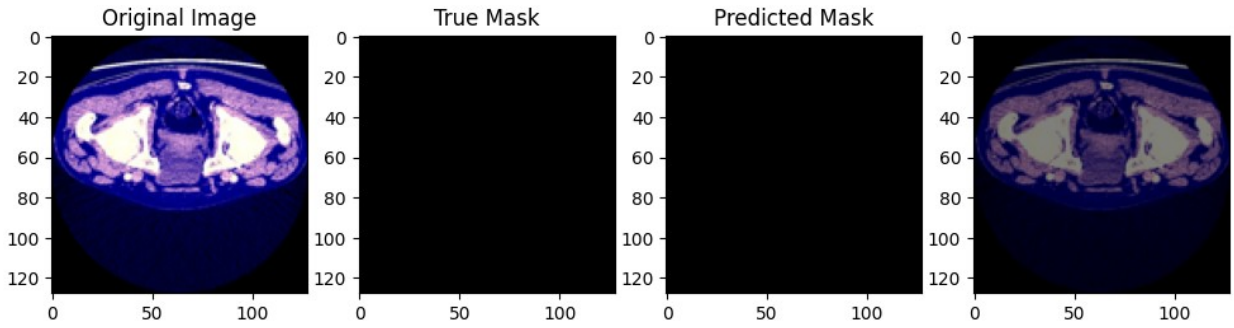

```
245
1/1 [==============================] - 0s 22ms/step
```



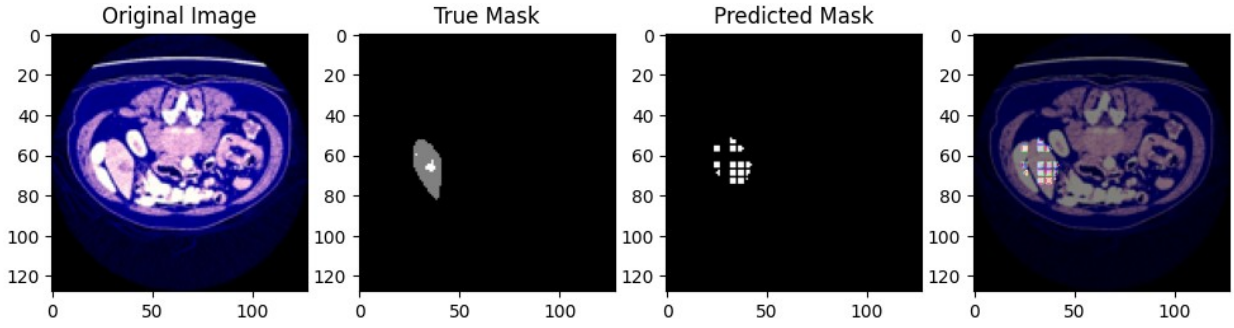

```
40
1/1 [==============================] - 0s 23ms/step
```



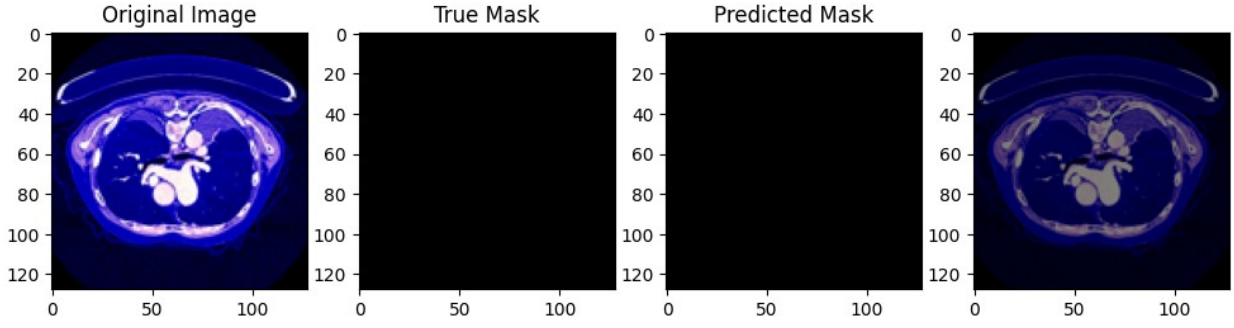

```
427
1/1 [==============================] - 0s 25ms/step
```



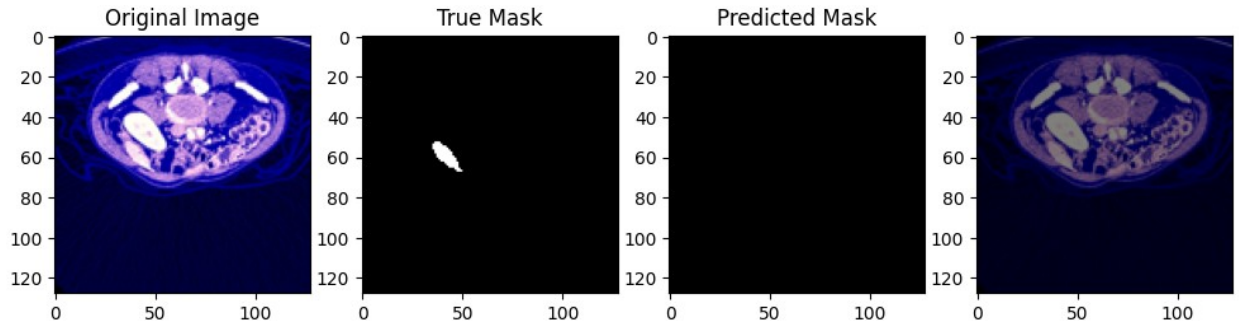

```
303
1/1 [==============================] - 0s 22ms/step
```



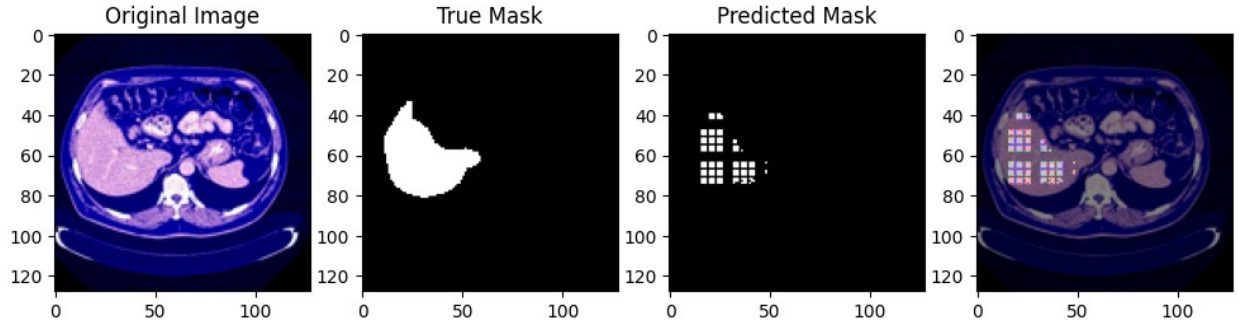

```
588
1/1 [==============================] - 0s 25ms/step
```



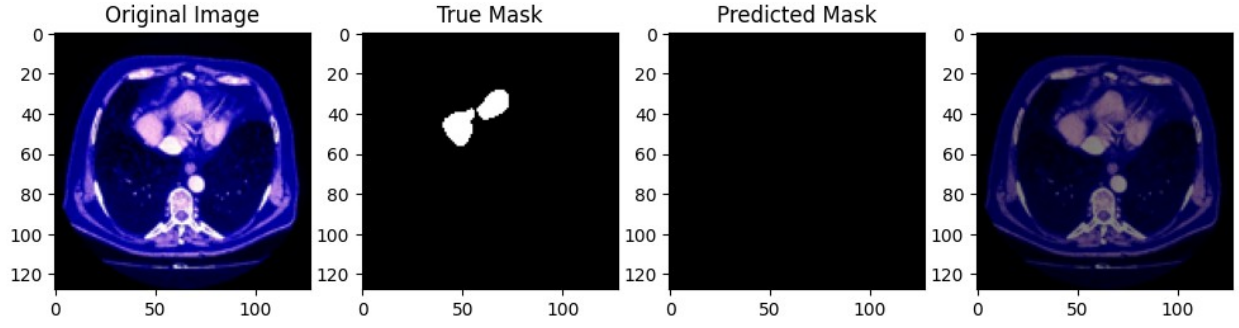

```
210
1/1 [==============================] - 0s 28ms/step
```



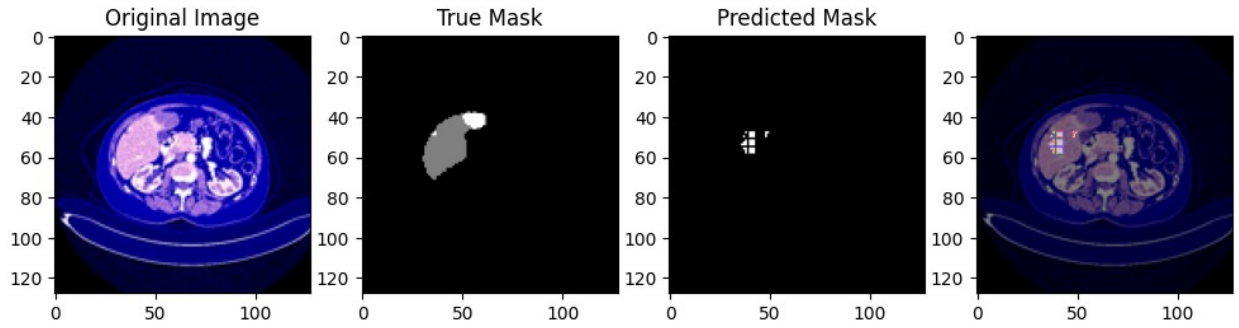

```
137
1/1 [==============================] - 0s 28ms/step
```



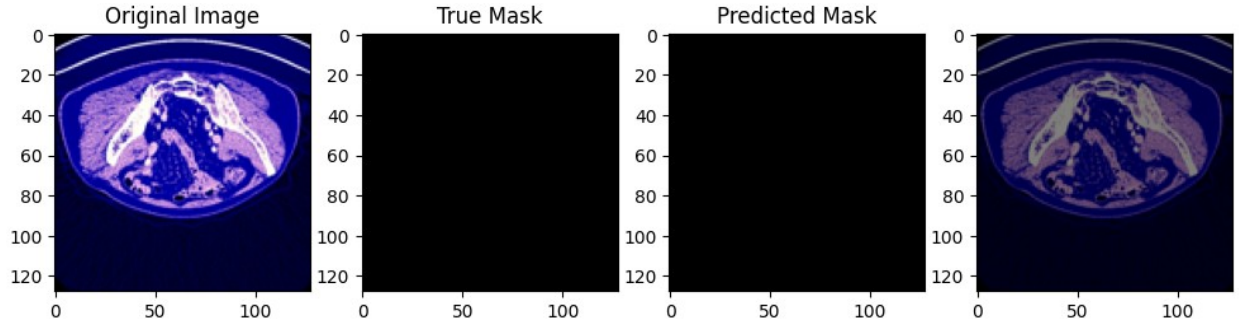

```
138
1/1 [==============================] - 0s 28ms/step
```



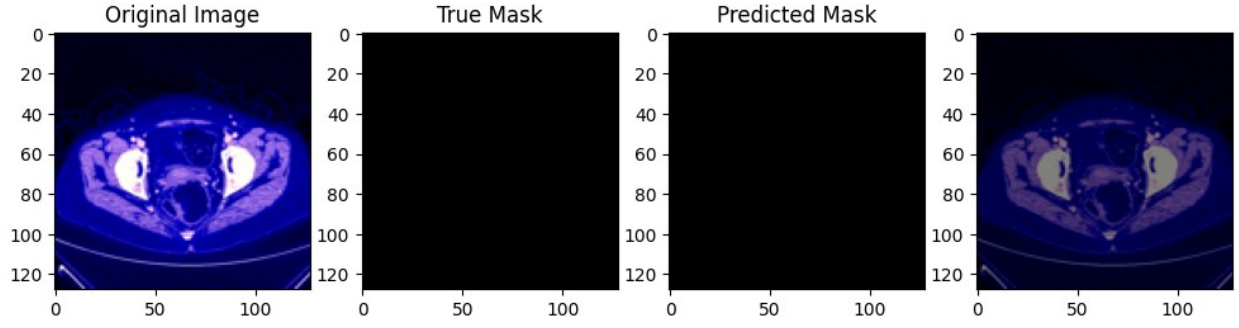

```
261
1/1 [==============================] - 0s 26ms/step
```



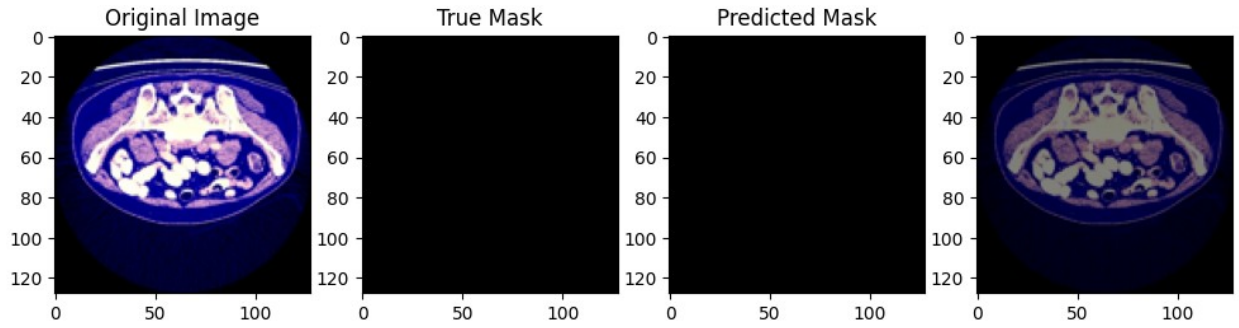

```
352
1/1 [==============================] - 0s 24ms/step
```



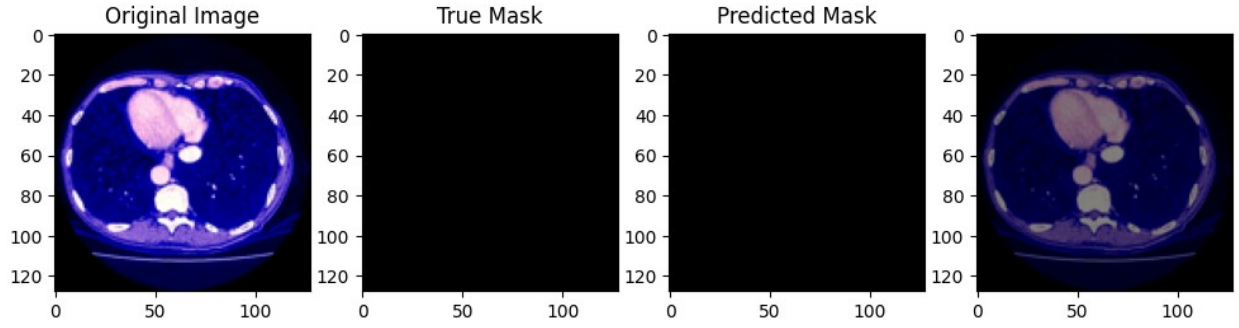

```
360
1/1 [==============================] - 0s 26ms/step
```



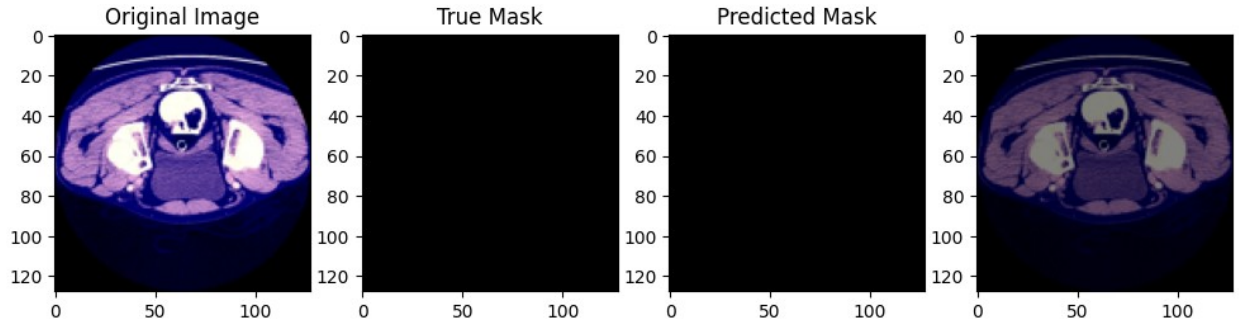

```
347
1/1 [==============================] - 0s 26ms/step
```



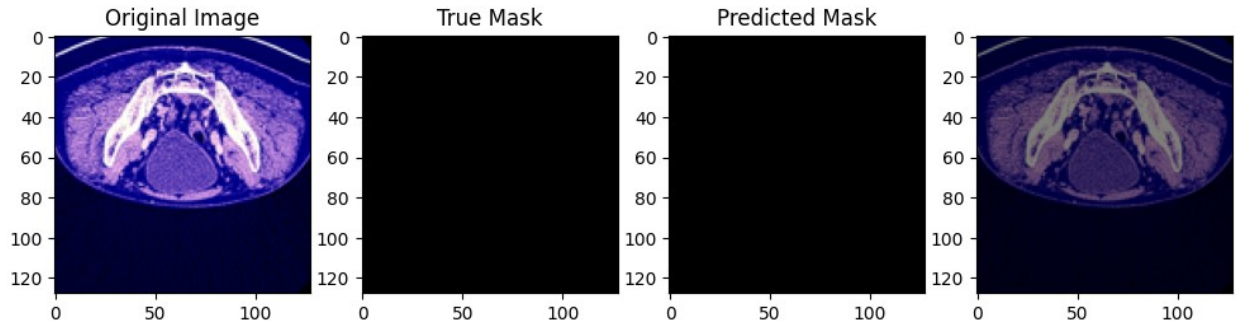

```
479
1/1 [==============================] - 0s 25ms/step
```



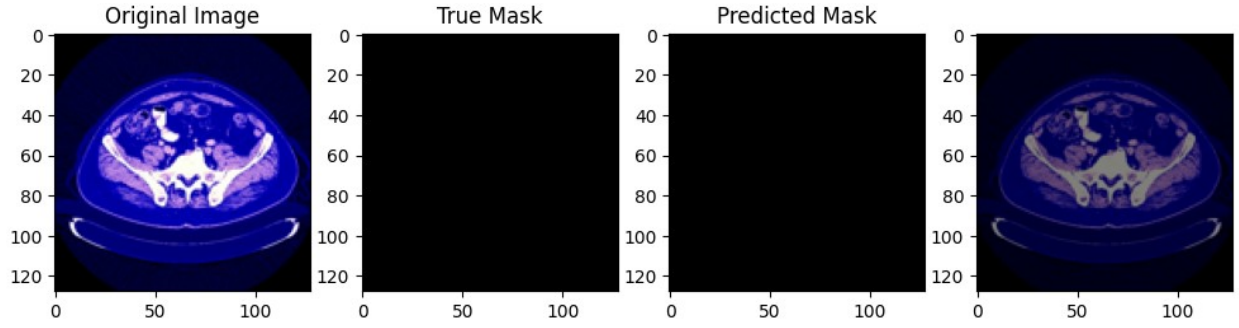

```
307
1/1 [==============================] - 0s 24ms/step
```



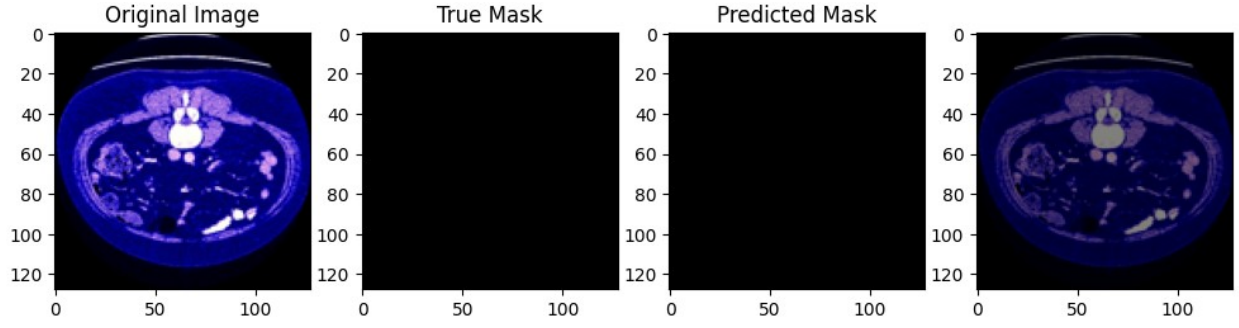

```
658
1/1 [==============================] - 0s 23ms/step
```



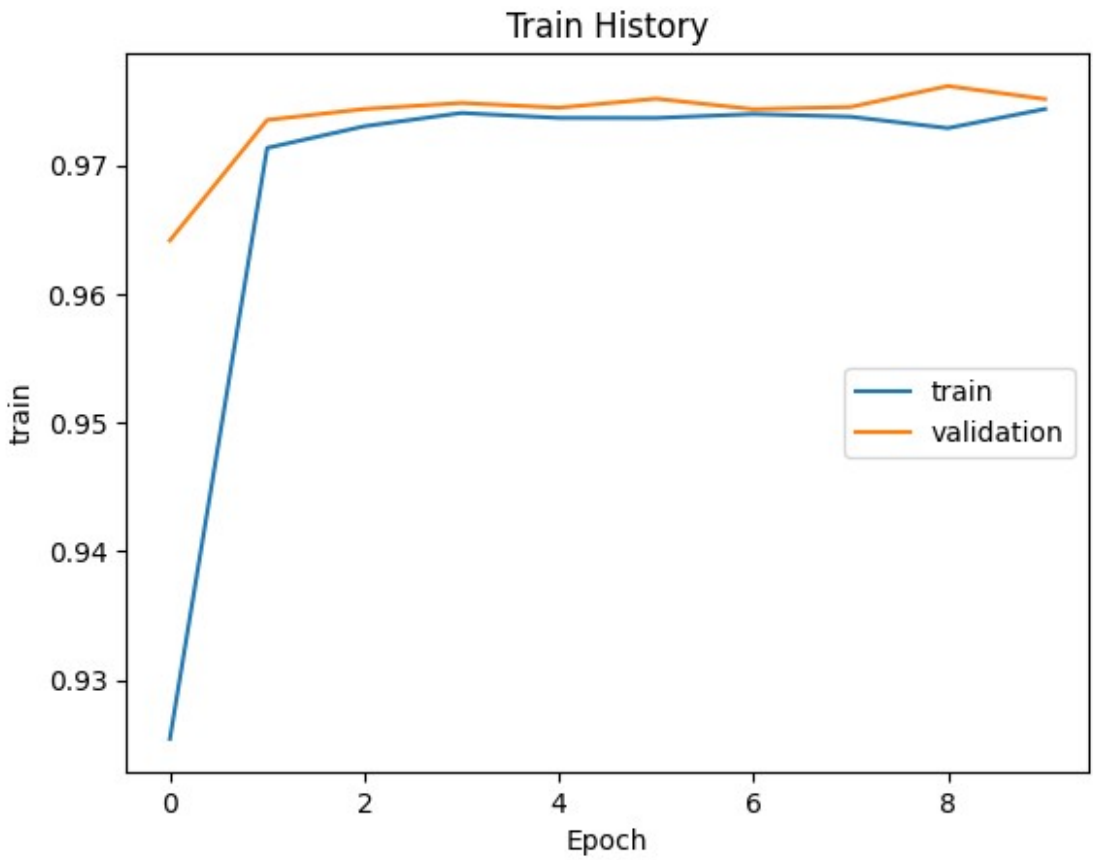

```
895
1/1 [==============================] - 0s 25ms/step
```



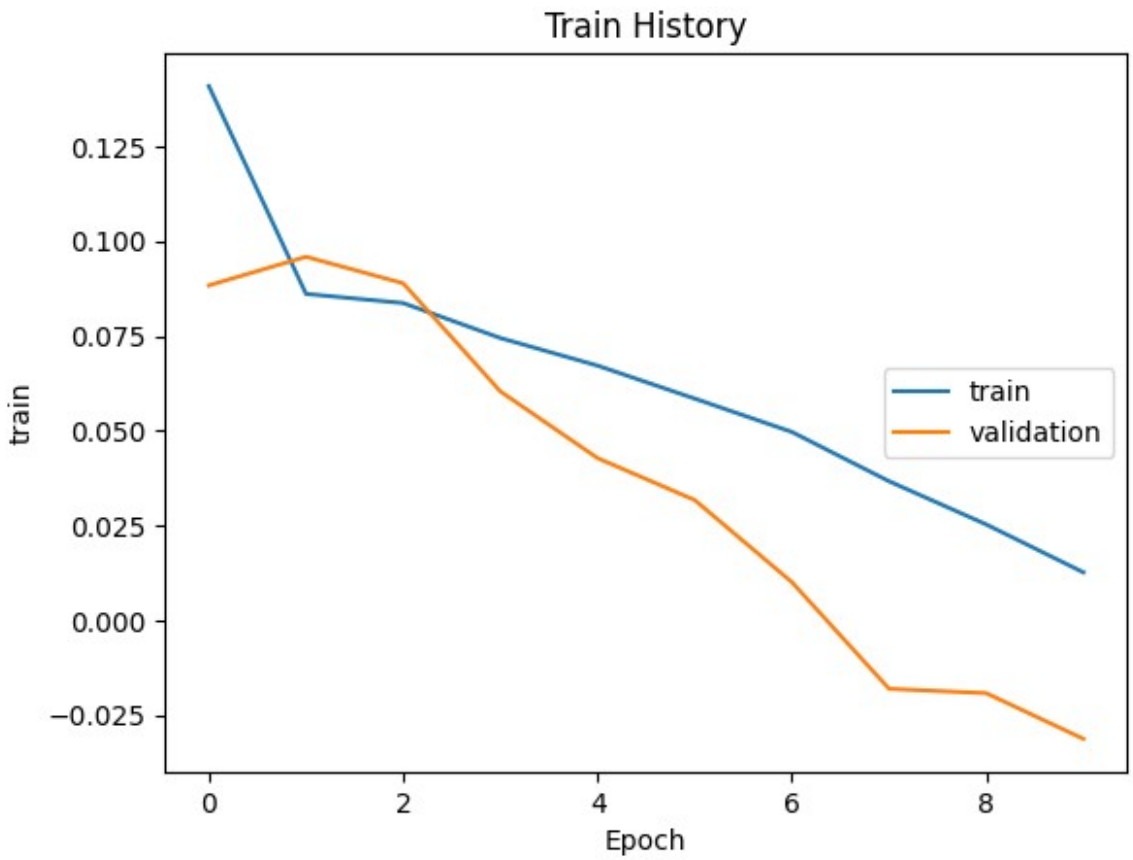

```
111
1/1 [==============================] - 0s 24ms/step
```



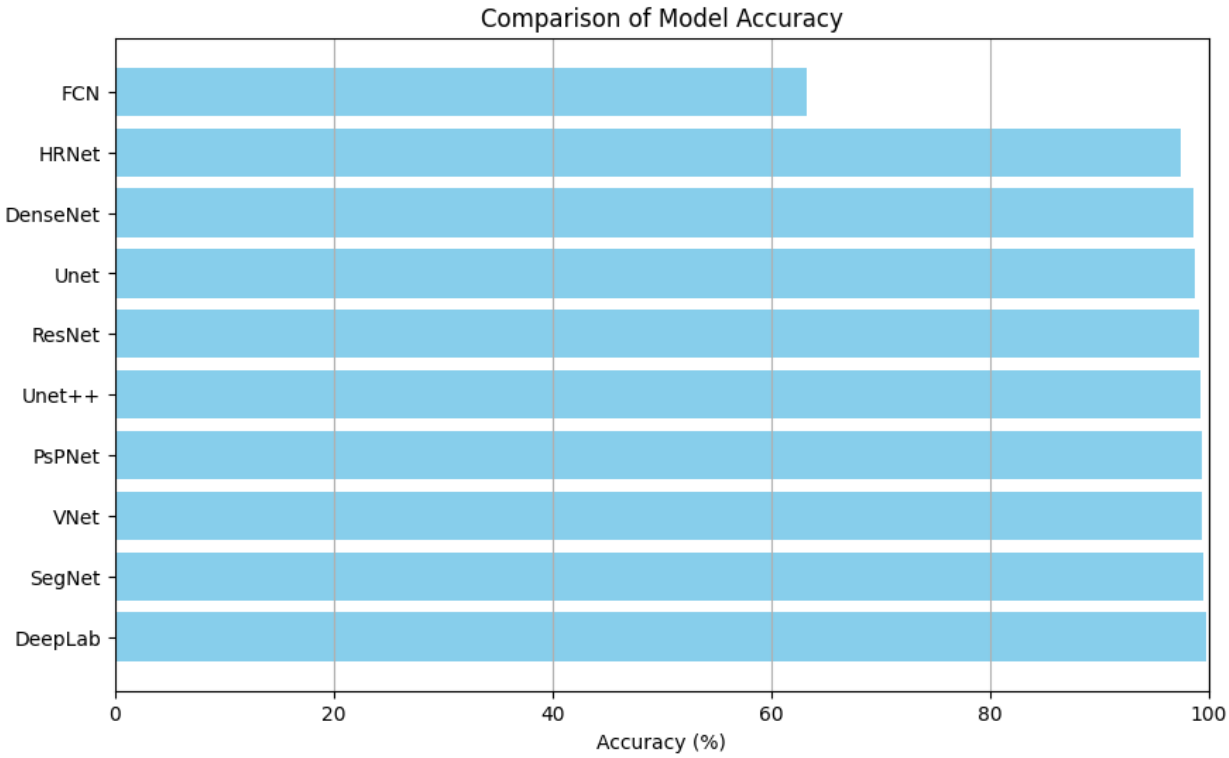

```
show_history(history9, 'dice_coefficient', 'val_dice_coefficient')
```




```
show_history(history9, 'loss', 'val_loss')
```




```python
import pandas as pd
import matplotlib.pyplot as plt

# Sample data (replace with your own)
models = ['FCN', 'Unet', 'SegNet', 'ResNet', 'PsPNet', 'DenseNet',
'Unet++', 'DeepLab', 'VNet', 'HRNet']
accuracy = [63.321, 98.783, 99.529, 99.166, 99.385, 98.57, 99.304,
99.818, 99.42, 97.51]  # Accuracy scores for each model

# Creating a DataFrame
data = pd.DataFrame({'Model': models, 'Accuracy': accuracy})

# Sorting models based on accuracy (optional)
data = data.sort_values(by='Accuracy', ascending=False)

# Plotting
plt.figure(figsize=(10, 6))
plt.barh(data['Model'], data['Accuracy'], color='skyblue')
plt.xlabel('Accuracy (%)')
plt.title('Comparison of Model Accuracy')
plt.xlim(0, 100)  # Optional, set the limit of x-axis
plt.grid(axis='x')  # Optional, add grid lines along x-axis
plt.show()
```


Comparison of Model Accuracy
FCN
HRNet
DenseNet
Unet
ResNet
Unet++
PsPNet
VNet
SegNet
DeepLab
0
20
40
60
80
100
Accuracy (%)