Crack Segmentation on deep crack dataset.

Importing the neccessary libraries and mounting the google drive

In [31]:
```
import os
import cv2
import shutil
import math
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
```

In [32]:
```
import tensorflow as tf
from tensorflow import keras
import tensorflow.keras.backend as K
from tensorflow.keras.utils import Sequence
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, Activa
from tensorflow.keras.losses import binary_crossentropy
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

In [33]:
```
from sklearn.metrics import classification_report, roc_auc_score, accuracy_sc
from albumentations import Compose, OneOf, Flip, Rotate, RandomContrast, Rand
```

In [34]:
```
from tensorflow.keras.losses import binary_crossentropy
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from skimage.transform import resize
from sklearn.metrics import classification_report
```

Loading the dataset

In [35]:
```
import os
import zipfile
```

Input Pipeline

- The data is loaded with its respective masks
- The data is then shuffled
- Splited & Divided into train,test and validation sets

In [36]:
```
train_image_dir = r'/home/ubuntu/Desktop/NNDL Project/train_img'
train_mask_dir = r'/home/ubuntu/Desktop/NNDL Project/train_lab'

test_image_dir = r'/home/ubuntu/Desktop/test_img'
test_mask_dir = r'/home/ubuntu/Desktop/test_lab'
```

In [37]:
```
train_image_paths = sorted([os.path.join(train_image_dir, fname) for fname in
train_mask_paths = sorted([os.path.join(train_mask_dir, fname) for fname in

test_image_paths = sorted([os.path.join(test_image_dir, fname) for fname in
```

```
test_mask_paths = sorted([os.path.join(test_mask_dir, fname) for fname in os.

print("Number of training images : ", len(train_image_paths))
print("Number of training masks : ", len(train_mask_paths))
print('\n')
print("Number of testing images : ", len(test_image_paths))
print("Number of testing masks : ", len(test_mask_paths))
```

```
Number of training images :  300
Number of training masks :  300


Number of testing images :  237
Number of testing masks :  237
```

In [38]:

```
# Shuffle
import random
combined = list(zip(train_image_paths, train_mask_paths))
random.shuffle(combined)

train_image_paths[:], train_mask_paths[:] = zip(*combined)
```

In [39]:

```
# Splitting
train_image_files = train_image_paths[:270]
train_mask_files = train_mask_paths[:270]

valid_image_files = train_image_paths[270:]
valid_mask_files = train_mask_paths[270:]

print(len(train_image_files), len(train_mask_files))
print(len(valid_image_files), len(valid_mask_files))
```

```
270 270
30 30
```

Generator creation

- A custom generator is created which combines the image with its repective masks.
- All data is augmented using albumentation library.
- Random flips and rotations along with more attributes of the albumentation library are used to augment the images present in the training data

In [40]:

```
batch_size = 10
img_dim=(256, 256)
```

In [41]:

```
class Generator(Sequence):

  def __init__(self, x_set, y_set, batch_size=5, img_dim=(128, 128), augment=
      self.x = x_set
      self.y = y_set
      self.batch_size = batch_size
      self.img_dim = img_dim
      self.augment = augment

  def __len__(self):
      return math.ceil(len(self.x) / self.batch_size)
```

```
  augmentations = Compose(
    [
      Flip(p=0.7),
      Rotate(p=0.7),
      OneOf([
              RandomContrast(),
              RandomGamma(),
              RandomBrightness()
            ], p=0.3),
      OneOf([
              ElasticTransform(alpha=120, sigma=120 * 0.05, alpha_affine=120
              GridDistortion(),
              OpticalDistortion(distort_limit=2, shift_limit=0.5)
            ], p=0.3),
    ])

  def __getitem__(self, idx):
      batch_x = self.x[idx * self.batch_size:(idx + 1) * self.batch_size]
      batch_y = self.y[idx * self.batch_size:(idx + 1) * self.batch_size]

      batch_x = np.array([cv2.resize(cv2.cvtColor(cv2.imread(file_name, -1),
      batch_y = np.array([(cv2.resize(cv2.imread(file_name, -1), (self.img_di

      if self.augment is True:
        aug = [self.augmentations(image=i, mask=j) for i, j in zip(batch_x, b
        batch_x = np.array([i['image'] for i in aug])
        batch_y = np.array([j['mask'] for j in aug])

      batch_y = np.expand_dims(batch_y, -1)

      return batch_x/255, batch_y/1
```

In [42]:

```
train_generator = Generator(train_image_files, train_mask_files)
validation_generator = Generator(valid_image_files, valid_mask_files)
test_generator=Generator(test_image_paths,test_mask_paths)
```

In [43]:

```
for i, j in train_generator:
  break

print(i.shape)
print(j.shape)
```

```
(5, 128, 128, 3)
(5, 128, 128, 1)
```

In [44]:

```
for i, j in validation_generator:
  break

print(i.shape)
print(j.shape)
```

```
(5, 128, 128, 3)
(5, 128, 128, 1)
```

In [45]:

```
for i,j in test_generator:
  break
print(i.shape)
print(j.shape)
```

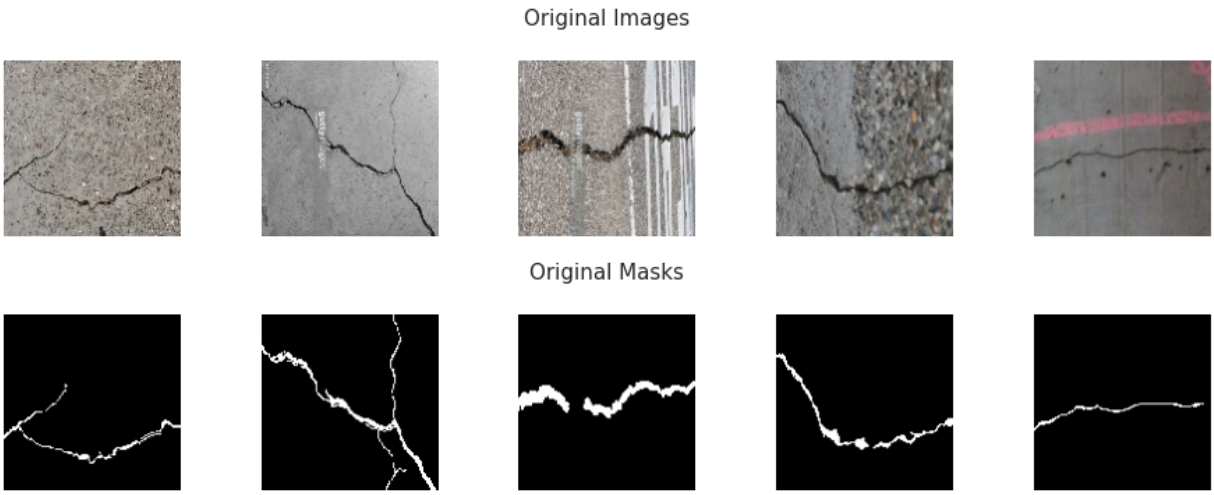
```
(5, 128, 128, 3)
```

```
(5, 128, 128, 1)
```

Plotting of the unaugmented images along with known images

In [46]:

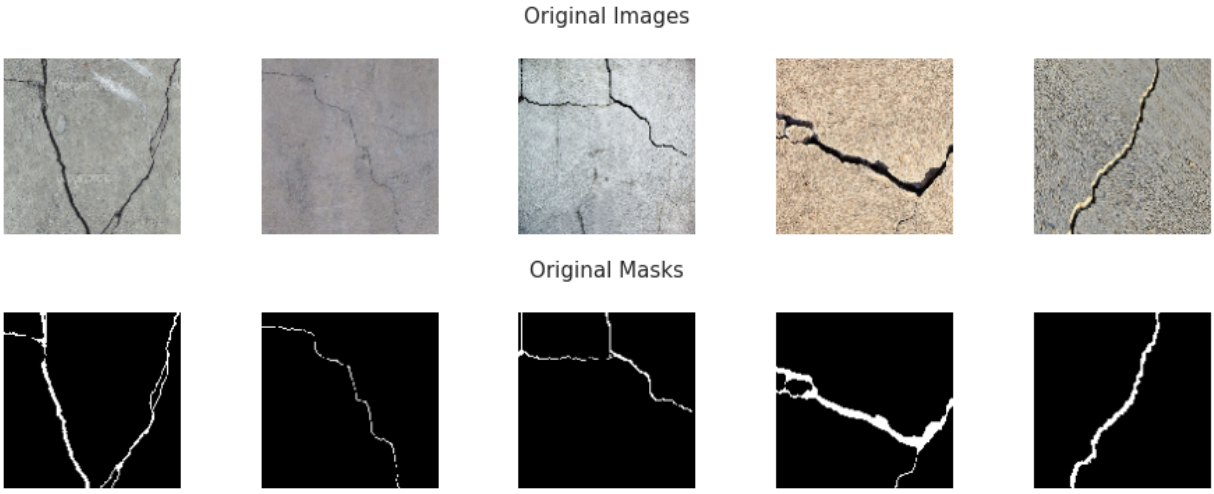
```
# Train generator samples (Un-augmented)
for i, j in train_generator:
    break

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Original Images', fontsize=15)
axes = axes.flatten()
for img, ax in zip(i[:5], axes[:5]):
    ax.imshow(img)
    ax.axis('off')
plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Original Masks', fontsize=15)
axes = axes.flatten()
for img, ax in zip(j[:5], axes[:5]):
    ax.imshow(np.squeeze(img, -1), cmap='gray')
    ax.axis('off')
plt.tight_layout()
plt.show()
```



In [47]:

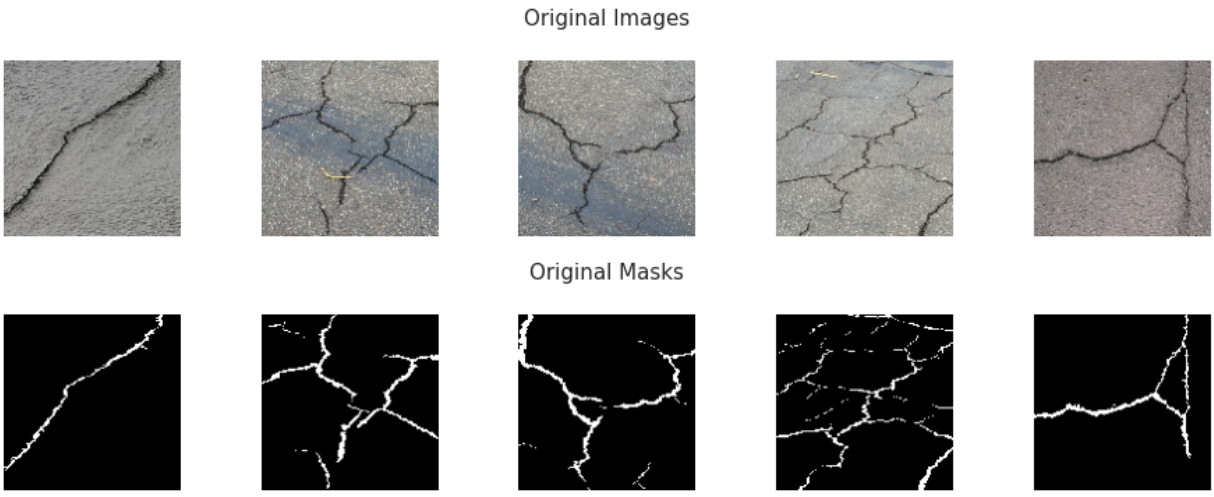
```
# Validation generator samples (Un-augmented)
for i, j in validation_generator:
    break

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Original Images', fontsize=15)
axes = axes.flatten()
for img, ax in zip(i[:5], axes[:5]):
    ax.imshow(img)
    ax.axis('off')
plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Original Masks', fontsize=15)
axes = axes.flatten()
for img, ax in zip(j[:5], axes[:5]):
    ax.imshow(np.squeeze(img, -1), cmap='gray')
    ax.axis('off')
```

```
plt.tight_layout()
plt.show()
```



In [48]:

```
# test generator samples (Un-augmented)
for i, j in test_generator:
    break

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Original Images', fontsize=15)
axes = axes.flatten()
for img, ax in zip(i[:5], axes[:5]):
    ax.imshow(img)
    ax.axis('off')
plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Original Masks', fontsize=15)
axes = axes.flatten()
for img, ax in zip(j[:5], axes[:5]):
    ax.imshow(np.squeeze(img, -1), cmap='gray')
    ax.axis('off')
plt.tight_layout()
plt.show()
```



In [49]:

```
tg = Generator(train_image_files, train_mask_files, batch_size, img_dim, augm
vg = Generator(valid_image_files, valid_mask_files, batch_size, img_dim, augm
testg=Generator(test_image_paths,test_mask_paths,batch_size,img_dim,augment=F
```

In [50]:
```
for i, j in tg:
  break

print(i.shape)
print(j.shape)
```

```
(10, 256, 256, 3)
(10, 256, 256, 1)
```

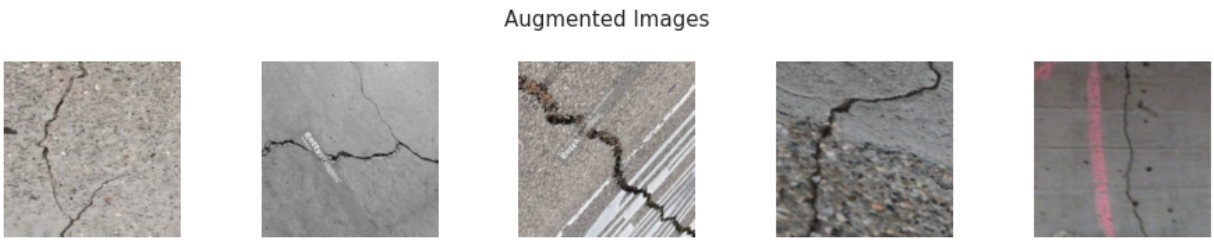
In [51]:
```
for i, j in vg:
  break

print(i.shape)
print(j.shape)
```

```
(10, 256, 256, 3)
(10, 256, 256, 1)
```

In [52]:
```
for i, j in testg:
  break

print(i.shape)
print(j.shape)
```

```
(10, 256, 256, 3)
(10, 256, 256, 1)
```

In [53]:
```
# Augmented train
for i, j in tg:
    break

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Augmented Images', fontsize=15)
axes = axes.flatten()
for img, ax in zip(i[:5], axes[:5]):
    ax.imshow(img)
    ax.axis('off')
plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, 5, figsize=(13,2.5))
fig.suptitle('Augmented Masks', fontsize=15)
axes = axes.flatten()
for img, ax in zip(j[:5], axes[:5]):
    ax.imshow(np.squeeze(img, -1), cmap='gray')
    ax.axis('off')
plt.tight_layout()
plt.show()
```



Augmented Masks

Model

In [54]:
```
import numpy as np
from tensorflow.keras.backend import int_shape
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, UpSampling2D, Add,
from tensorflow.keras.regularizers import l2
```

In [55]:
```
# BatchNormalization and Activation
def BN_Act(x, act = True):
    x = BatchNormalization()(x)
    if act == True:
        x = Activation("relu")(x)
    return x
```

In [56]:
```
# conv2d block
def conv2d_block(x, filters, kernel_size = (3, 3), padding = "same", strides
    conv = BN_Act(x)
    conv = Conv2D(filters, kernel_size, padding = padding, strides = strides)
    return conv
```

In [57]:
```
# Fixed layer
def stem(x, filters, kernel_size=(3, 3), padding="same", strides=1):
    conv = Conv2D(filters, kernel_size, padding = padding, strides = strides)
    conv = conv2d_block(conv, filters, kernel_size = kernel_size, padding = p

    #skip
    shortcut = Conv2D(filters, kernel_size = (1, 1), padding = padding, stric
    shortcut = BN_Act(shortcut, act = False) # No activation in skip connecti

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

In [58]:
```
#Residual Block
def residual_block(x, filters, kernel_size = (3, 3), padding = "same", stride
    res = conv2d_block(x, filters, kernel_size = kernel_size, padding = paddi
    res = conv2d_block(res, filters, kernel_size = kernel_size, padding = pac

    shortcut = Conv2D(filters, kernel_size = (1, 1), padding = padding, stric
    shortcut = BN_Act(shortcut, act = False) # No activation in skip connecti

    output = Add()([shortcut, res])
    return output
```

In [59]:
```
#Upsample concatenation block
def upsample_concat_block(x, xskip):
```

```
    u = UpSampling2D((2, 2))(x)
    c = Concatenate()([u, xskip])
    return c
```

In [60]:

```
# Complete model Architecture
def ResUNet():
    f = [16, 32, 64, 128, 256]
    inputs = Input((img_dim[0], img_dim[1], 3))

    ## Encoder/downsampling/contracting path
    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/Bottleneck
    b0 = conv2d_block(e5, f[4], strides = 1)
    b1 = conv2d_block(b0, f[4], strides = 1)

    ## Decoder/upsampling/expansive path
    u1 = upsample_concat_block(b1, e4)
    d1 = residual_block(u1, f[4])

    u2 = upsample_concat_block(d1, e3)
    d2 = residual_block(u2, f[3])

    u3 = upsample_concat_block(d2, e2)
    d3 = residual_block(u3, f[2])

    u4 = upsample_concat_block(d3, e1)
    d4 = residual_block(u4, f[1])

    outputs = Conv2D(1, (1, 1), padding = "same", activation = "sigmoid")(d4)
    model = Model(inputs, outputs)
    return model
```

In [61]:

```
K.clear_session()
model = ResUNet()
```

```
2022-04-01 22:11:16.439548: I tensorflow/compiler/jit/xla_cpu_device.cc:41] N
ot creating XLA devices, tf_xla_enable_xla_devices not set
2022-04-01 22:11:16.439663: I tensorflow/core/platform/cpu_feature_guard.cc:1
42] This TensorFlow binary is optimized with oneAPI Deep Neural Network Libra
ry (oneDNN) to use the following CPU instructions in performance-critical ope
rations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate c
ompiler flags.
2022-04-01 22:11:16.440057: I tensorflow/core/common_runtime/process_util.cc:
146] Creating new thread pool with default inter op setting: 2. Tune using in
ter_op_parallelism_threads for best performance.
```

In [62]:

```
model.summary()
```

```
Model: "model"
_________________________________________________________________________
_________________
Layer (type)                 Output Shape              Param #     Connected to
=========================================================================
```

```
==========
input_1 (InputLayer)            [(None, 256, 256, 3) 0
_____________________
conv2d (Conv2D)                 (None, 256, 256, 16) 448         input_1[0]
[0]
_____________________
batch_normalization (BatchNorma (None, 256, 256, 16) 64          conv2d[0][0]
_____________________
activation (Activation)         (None, 256, 256, 16) 0           batch_normal
ization[0][0]
_____________________
conv2d_2 (Conv2D)               (None, 256, 256, 16) 64          input_1[0]
[0]
_____________________
conv2d_1 (Conv2D)               (None, 256, 256, 16) 2320        activation
[0][0]
_____________________
batch_normalization_1 (BatchNor (None, 256, 256, 16) 64          conv2d_2[0]
[0]
_____________________
add (Add)                       (None, 256, 256, 16) 0           conv2d_1[0]
[0]
                                                                 batch_normal

ization_1[0][0]
_____________________
batch_normalization_2 (BatchNor (None, 256, 256, 16) 64          add[0][0]
_____________________
activation_1 (Activation)       (None, 256, 256, 16) 0           batch_normal
ization_2[0][0]
_____________________
conv2d_3 (Conv2D)               (None, 128, 128, 32) 4640        activation_1
[0][0]
_____________________
batch_normalization_3 (BatchNor (None, 128, 128, 32) 128         conv2d_3[0]
[0]
_____________________
conv2d_5 (Conv2D)               (None, 128, 128, 32) 544         add[0][0]
_____________________
activation_2 (Activation)       (None, 128, 128, 32) 0           batch_normal
ization_3[0][0]
_____________________
batch_normalization_4 (BatchNor (None, 128, 128, 32) 128         conv2d_5[0]
[0]
_____________________
conv2d_4 (Conv2D)               (None, 128, 128, 32) 9248        activation_2
[0][0]
_____________________
add_1 (Add)                     (None, 128, 128, 32) 0           batch_normal
```

```
ization_4[0][0]
                                                                 conv2d_4[0]
[0]
__________________________________________________________________________
_____________________
batch_normalization_5 (BatchNor (None, 128, 128, 32) 128         add_1[0][0]
__________________________________________________________________________
_____________________
activation_3 (Activation)       (None, 128, 128, 32) 0           batch_normal
ization_5[0][0]
__________________________________________________________________________
_____________________
conv2d_6 (Conv2D)               (None, 64, 64, 64)   18496       activation_3
[0][0]
__________________________________________________________________________
_____________________
batch_normalization_6 (BatchNor (None, 64, 64, 64)   256         conv2d_6[0]
[0]
__________________________________________________________________________
_____________________
conv2d_8 (Conv2D)               (None, 64, 64, 64)   2112        add_1[0][0]
__________________________________________________________________________
_____________________
activation_4 (Activation)       (None, 64, 64, 64)   0           batch_normal
ization_6[0][0]
__________________________________________________________________________
_____________________
batch_normalization_7 (BatchNor (None, 64, 64, 64)   256         conv2d_8[0]
[0]
__________________________________________________________________________
_____________________
conv2d_7 (Conv2D)               (None, 64, 64, 64)   36928       activation_4
[0][0]
__________________________________________________________________________
_____________________
add_2 (Add)                     (None, 64, 64, 64)   0           batch_normal
ization_7[0][0]
                                                                 conv2d_7[0]
[0]
__________________________________________________________________________
_____________________
batch_normalization_8 (BatchNor (None, 64, 64, 64)   256         add_2[0][0]
__________________________________________________________________________
_____________________
activation_5 (Activation)       (None, 64, 64, 64)   0           batch_normal
ization_8[0][0]
__________________________________________________________________________
_____________________
conv2d_9 (Conv2D)               (None, 32, 32, 128)  73856       activation_5
[0][0]
__________________________________________________________________________
_____________________
batch_normalization_9 (BatchNor (None, 32, 32, 128)  512         conv2d_9[0]
[0]
__________________________________________________________________________
_____________________
conv2d_11 (Conv2D)              (None, 32, 32, 128)  8320        add_2[0][0]
__________________________________________________________________________
_____________________
activation_6 (Activation)       (None, 32, 32, 128)  0           batch_normal
ization_9[0][0]
__________________________________________________________________________
_____________________
batch_normalization_10 (BatchNo (None, 32, 32, 128)  512         conv2d_11[0]
```

```
[0]
__________________________________________________________________________________________________
__________________________
conv2d_10 (Conv2D)              (None, 32, 32, 128)  147584      activation_6
[0][0]
__________________________________________________________________________________________________
__________________________
add_3 (Add)                     (None, 32, 32, 128)  0           batch_normal
ization_10[0][0]
                                                                 conv2d_10[0]
[0]
__________________________________________________________________________________________________
__________________________
batch_normalization_11 (BatchNo (None, 32, 32, 128)  512         add_3[0][0]
__________________________________________________________________________________________________
__________________________
activation_7 (Activation)       (None, 32, 32, 128)  0           batch_normal
ization_11[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_12 (Conv2D)              (None, 16, 16, 256)  295168      activation_7
[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_12 (BatchNo (None, 16, 16, 256)  1024        conv2d_12[0]
[0]
__________________________________________________________________________________________________
__________________________
conv2d_14 (Conv2D)              (None, 16, 16, 256)  33024       add_3[0][0]
__________________________________________________________________________________________________
__________________________
activation_8 (Activation)       (None, 16, 16, 256)  0           batch_normal
ization_12[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_13 (BatchNo (None, 16, 16, 256)  1024        conv2d_14[0]
[0]
__________________________________________________________________________________________________
__________________________
conv2d_13 (Conv2D)              (None, 16, 16, 256)  590080      activation_8
[0][0]
__________________________________________________________________________________________________
__________________________
add_4 (Add)                     (None, 16, 16, 256)  0           batch_normal
ization_13[0][0]
                                                                 conv2d_13[0]
[0]
__________________________________________________________________________________________________
__________________________
batch_normalization_14 (BatchNo (None, 16, 16, 256)  1024        add_4[0][0]
__________________________________________________________________________________________________
__________________________
activation_9 (Activation)       (None, 16, 16, 256)  0           batch_normal
ization_14[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_15 (Conv2D)              (None, 16, 16, 256)  590080      activation_9
[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_15 (BatchNo (None, 16, 16, 256)  1024        conv2d_15[0]
[0]
__________________________________________________________________________________________________
__________________________
```

```
activation_10 (Activation)      (None, 16, 16, 256)  0           batch_normal
ization_15[0][0]
__________________________________________________________________________________
_____________________
conv2d_16 (Conv2D)              (None, 16, 16, 256)  590080      activation_1
0[0][0]
__________________________________________________________________________________
_____________________
up_sampling2d (UpSampling2D)    (None, 32, 32, 256)  0           conv2d_16[0]
[0]
__________________________________________________________________________________
_____________________
concatenate (Concatenate)       (None, 32, 32, 384)  0           up_sampling2
d[0][0]
                                                                 add_3[0][0]
__________________________________________________________________________________
_____________________
batch_normalization_16 (BatchNo (None, 32, 32, 384)  1536        concatenate
[0][0]
__________________________________________________________________________________
_____________________
activation_11 (Activation)      (None, 32, 32, 384)  0           batch_normal
ization_16[0][0]
__________________________________________________________________________________
_____________________
conv2d_17 (Conv2D)              (None, 32, 32, 256)  884992      activation_1
1[0][0]
__________________________________________________________________________________
_____________________
batch_normalization_17 (BatchNo (None, 32, 32, 256)  1024        conv2d_17[0]
[0]
__________________________________________________________________________________
_____________________
conv2d_19 (Conv2D)              (None, 32, 32, 256)  98560       concatenate
[0][0]
__________________________________________________________________________________
_____________________
activation_12 (Activation)      (None, 32, 32, 256)  0           batch_normal
ization_17[0][0]
__________________________________________________________________________________
_____________________
batch_normalization_18 (BatchNo (None, 32, 32, 256)  1024        conv2d_19[0]
[0]
__________________________________________________________________________________
_____________________
conv2d_18 (Conv2D)              (None, 32, 32, 256)  590080      activation_1
2[0][0]
__________________________________________________________________________________
_____________________
add_5 (Add)                     (None, 32, 32, 256)  0           batch_normal
ization_18[0][0]
                                                                 conv2d_18[0]
[0]
__________________________________________________________________________________
_____________________
up_sampling2d_1 (UpSampling2D)  (None, 64, 64, 256)  0           add_5[0][0]
__________________________________________________________________________________
_____________________
concatenate_1 (Concatenate)     (None, 64, 64, 320)  0           up_sampling2
d_1[0][0]
                                                                 add_2[0][0]
__________________________________________________________________________________
_____________________
batch_normalization_19 (BatchNo (None, 64, 64, 320)  1280        concatenate_
```

```
1[0][0]
__________________________________________________________________________________________________
__________________________
activation_13 (Activation)      (None, 64, 64, 320)  0           batch_normal
ization_19[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_20 (Conv2D)              (None, 64, 64, 128)  368768      activation_1
3[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_20 (BatchNo (None, 64, 64, 128)  512         conv2d_20[0]
[0]
__________________________________________________________________________________________________
__________________________
conv2d_22 (Conv2D)              (None, 64, 64, 128)  41088       concatenate_
1[0][0]
__________________________________________________________________________________________________
__________________________
activation_14 (Activation)      (None, 64, 64, 128)  0           batch_normal
ization_20[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_21 (BatchNo (None, 64, 64, 128)  512         conv2d_22[0]
[0]
__________________________________________________________________________________________________
__________________________
conv2d_21 (Conv2D)              (None, 64, 64, 128)  147584      activation_1
4[0][0]
__________________________________________________________________________________________________
__________________________
add_6 (Add)                     (None, 64, 64, 128)  0           batch_normal
ization_21[0][0]
                                                                 conv2d_21[0]
[0]
__________________________________________________________________________________________________
__________________________
up_sampling2d_2 (UpSampling2D)  (None, 128, 128, 128 0           add_6[0][0]
__________________________________________________________________________________________________
__________________________
concatenate_2 (Concatenate)     (None, 128, 128, 160 0           up_sampling2
d_2[0][0]
                                                                 add_1[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_22 (BatchNo (None, 128, 128, 160 640         concatenate_
2[0][0]
__________________________________________________________________________________________________
__________________________
activation_15 (Activation)      (None, 128, 128, 160 0           batch_normal
ization_22[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_23 (Conv2D)              (None, 128, 128, 64) 92224       activation_1
5[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_23 (BatchNo (None, 128, 128, 64) 256         conv2d_23[0]
[0]
__________________________________________________________________________________________________
__________________________
conv2d_25 (Conv2D)              (None, 128, 128, 64) 10304       concatenate_
2[0][0]
__________________________________________________________________________________________________
```

```
__________________________
activation_16 (Activation)      (None, 128, 128, 64) 0           batch_normal
ization_23[0][0]
__________________________________________________________________________
__________________________
batch_normalization_24 (BatchNo (None, 128, 128, 64) 256         conv2d_25[0]
[0]
__________________________________________________________________________
__________________________
conv2d_24 (Conv2D)              (None, 128, 128, 64) 36928       activation_1
6[0][0]
__________________________________________________________________________
__________________________
add_7 (Add)                     (None, 128, 128, 64) 0           batch_normal
ization_24[0][0]
                                                                 conv2d_24[0]
[0]
__________________________________________________________________________
__________________________
up_sampling2d_3 (UpSampling2D)  (None, 256, 256, 64) 0           add_7[0][0]
__________________________________________________________________________
__________________________
concatenate_3 (Concatenate)     (None, 256, 256, 80) 0           up_sampling2
d_3[0][0]
                                                                 add[0][0]
__________________________________________________________________________
__________________________
batch_normalization_25 (BatchNo (None, 256, 256, 80) 320         concatenate_
3[0][0]
__________________________________________________________________________
__________________________
activation_17 (Activation)      (None, 256, 256, 80) 0           batch_normal
ization_25[0][0]
__________________________________________________________________________
__________________________
conv2d_26 (Conv2D)              (None, 256, 256, 32) 23072       activation_1
7[0][0]
__________________________________________________________________________
__________________________
batch_normalization_26 (BatchNo (None, 256, 256, 32) 128         conv2d_26[0]
[0]
__________________________________________________________________________
__________________________
conv2d_28 (Conv2D)              (None, 256, 256, 32) 2592        concatenate_
3[0][0]
__________________________________________________________________________
__________________________
activation_18 (Activation)      (None, 256, 256, 32) 0           batch_normal
ization_26[0][0]
__________________________________________________________________________
__________________________
batch_normalization_27 (BatchNo (None, 256, 256, 32) 128         conv2d_28[0]
[0]
__________________________________________________________________________
__________________________
conv2d_27 (Conv2D)              (None, 256, 256, 32) 9248        activation_1
8[0][0]
__________________________________________________________________________
__________________________
add_8 (Add)                     (None, 256, 256, 32) 0           batch_normal
ization_27[0][0]
                                                                 conv2d_27[0]
[0]
__________________________________________________________________________
```

```
__________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 256, 256, 1)  33          add_8[0][0]
==================================================================================
======================
Total params: 4,723,057
Trainable params: 4,715,761
Non-trainable params: 7,296
__________________________________________________________________________________
_____________________
```

Loss Function

In [63]:

```
smooth = 1.

def dice_coef(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = tf.reduce_sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (tf.reduce_sum(y_true_f) + tf.reduc


def dice_coef_loss(y_true, y_pred):
    return 1.0 - dice_coef(y_true, y_pred)

def IOU(y_true, y_pred):

    y_true = K.flatten(y_true)
    y_pred = K.flatten(y_pred)

    thresh = 0.5

    y_true = K.cast(K.greater_equal(y_true, thresh), 'float32')
    y_pred = K.cast(K.greater_equal(y_pred, thresh), 'float32')

    union = K.sum(K.maximum(y_true, y_pred)) + K.epsilon()
    intersection = K.sum(K.minimum(y_true, y_pred)) + K.epsilon()

    iou = intersection/union

    return iou

# def bce_dice_loss(y_true, y_pred):

#     return binary_crossentropy(y_true, y_pred) + dice_loss(y_true, y_pred)
```

Learning Rate Schedular And other callbacks

In [64]:

```
def lr_schedule(epoch):
    lr =0.001
    if epoch >150:
        lr *=2**-1
    elif epoch >80:
        lr *=2**(-1)
    elif epoch >50:
        lr *=2**(-1)
    elif epoch >30:
        lr *=2**(-1)
```

```
    print('Learning rate: ', lr)
    return lr
```

In [65]:
```
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.optimizers import SGD
```

In [66]:
```
import time

start_time = time.time()

# Prepare callbacks for model saving and for learning rate adjustment.
lr_scheduler = LearningRateScheduler(lr_schedule)

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               min_lr=0.5e-8)

callbacks = [lr_reducer, lr_scheduler]
```

Compiling with adam and beyond

In [67]:
```
import tensorflow as tf
learning_rate = 0.0035
optimiser=tf.keras.optimizers.Adam(
    learning_rate=lr_schedule(0),
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=True,
    name="Adam"
)
model.compile(optimizer =optimiser , loss = dice_coef_loss, metrics = ['accur
```

```
Learning rate:  0.001
```

In [68]:
```
# model.save('/content/drive/My Drive/Colab Notebooks/Adam_and_beyond.h5')
```

Training

In [69]:
```
train_steps = len(train_image_files)//batch_size
valid_steps = len(valid_image_files)//batch_size

history = model.fit(
    tg,
    steps_per_epoch=train_steps,
    initial_epoch = 0,
    epochs=100,
    validation_data = vg,
    validation_steps = valid_steps,callbacks=callbacks)
```

```
2022-04-01 22:13:17.510613: I tensorflow/compiler/mlir/mlir_graph_optimizatio
n_pass.cc:116] None of the MLIR optimization passes are enabled (registered
2)
```

```
2022-04-01 22:13:17.530562: I tensorflow/core/platform/profile_utils/cpu_util
s.cc:112] CPU Frequency: 3799900000 Hz
Epoch 1/100
Learning rate:  0.001
27/27 [==============================] - 196s 7s/step - loss: 0.6930 - accura
cy: 0.8001 - IOU: 0.2422 - dice_coef: 0.3070 - val_loss: 1.0000 - val_accurac
y: 0.9601 - val_IOU: 4.1994e-12 - val_dice_coef: 4.2034e-05
Epoch 2/100
Learning rate:  0.001
27/27 [==============================] - 196s 7s/step - loss: 0.3007 - accura
cy: 0.9804 - IOU: 0.5481 - dice_coef: 0.6993 - val_loss: 0.9283 - val_accurac
y: 0.5345 - val_IOU: 0.0366 - val_dice_coef: 0.0717
Epoch 3/100
Learning rate:  0.001
27/27 [==============================] - 190s 7s/step - loss: 0.2588 - accura
cy: 0.9818 - IOU: 0.5953 - dice_coef: 0.7412 - val_loss: 0.9476 - val_accurac
y: 0.8951 - val_IOU: 0.0268 - val_dice_coef: 0.0524
Epoch 4/100
Learning rate:  0.001
27/27 [==============================] - 193s 7s/step - loss: 0.2403 - accura
cy: 0.9843 - IOU: 0.6188 - dice_coef: 0.7597 - val_loss: 0.9999 - val_accurac
y: 0.9601 - val_IOU: 4.1994e-12 - val_dice_coef: 1.2708e-04
Epoch 5/100
Learning rate:  0.001
27/27 [==============================] - 190s 7s/step - loss: 0.2191 - accura
cy: 0.9845 - IOU: 0.6468 - dice_coef: 0.7809 - val_loss: 0.9176 - val_accurac
y: 0.1938 - val_IOU: 0.0431 - val_dice_coef: 0.0824
Epoch 6/100
Learning rate:  0.001
27/27 [==============================] - 187s 7s/step - loss: 0.2293 - accura
cy: 0.9853 - IOU: 0.6331 - dice_coef: 0.7707 - val_loss: 0.7497 - val_accurac
y: 0.9644 - val_IOU: 0.1454 - val_dice_coef: 0.2503
Epoch 7/100
Learning rate:  0.001
27/27 [==============================] - 187s 7s/step - loss: 0.2303 - accura
cy: 0.9837 - IOU: 0.6321 - dice_coef: 0.7697 - val_loss: 0.9129 - val_accurac
y: 0.9612 - val_IOU: 0.0387 - val_dice_coef: 0.0871
Epoch 8/100
Learning rate:  0.001
27/27 [==============================] - 186s 7s/step - loss: 0.2288 - accura
cy: 0.9847 - IOU: 0.6324 - dice_coef: 0.7712 - val_loss: 0.5708 - val_accurac
y: 0.9687 - val_IOU: 0.2875 - val_dice_coef: 0.4292
Epoch 9/100
Learning rate:  0.001
27/27 [==============================] - 188s 7s/step - loss: 0.2040 - accura
cy: 0.9856 - IOU: 0.6665 - dice_coef: 0.7960 - val_loss: 0.9906 - val_accurac
y: 0.9602 - val_IOU: 0.0042 - val_dice_coef: 0.0094
Epoch 10/100
Learning rate:  0.001
27/27 [==============================] - 186s 7s/step - loss: 0.2029 - accura
cy: 0.9861 - IOU: 0.6679 - dice_coef: 0.7971 - val_loss: 0.6569 - val_accurac
y: 0.9671 - val_IOU: 0.2221 - val_dice_coef: 0.3431
Epoch 11/100
Learning rate:  0.001
27/27 [==============================] - 187s 7s/step - loss: 0.1915 - accura
cy: 0.9869 - IOU: 0.6829 - dice_coef: 0.8085 - val_loss: 0.8214 - val_accurac
y: 0.9632 - val_IOU: 0.1049 - val_dice_coef: 0.1786
Epoch 12/100
Learning rate:  0.001
27/27 [==============================] - 185s 7s/step - loss: 0.2108 - accura
cy: 0.9853 - IOU: 0.6559 - dice_coef: 0.7892 - val_loss: 0.9398 - val_accurac
y: 0.9611 - val_IOU: 0.0317 - val_dice_coef: 0.0602
Epoch 13/100
Learning rate:  0.001
```

```
27/27 [==============================] - 187s 7s/step - loss: 0.2133 - accura
cy: 0.9862 - IOU: 0.6540 - dice_coef: 0.7867 - val_loss: 0.5702 - val_accurac
y: 0.9694 - val_IOU: 0.2866 - val_dice_coef: 0.4298
Epoch 14/100
Learning rate:  0.001
27/27 [==============================] - 188s 7s/step - loss: 0.2444 - accura
cy: 0.9836 - IOU: 0.6142 - dice_coef: 0.7556 - val_loss: 0.3949 - val_accurac
y: 0.9748 - val_IOU: 0.4477 - val_dice_coef: 0.6051
Epoch 15/100
Learning rate:  0.001
27/27 [==============================] - 189s 7s/step - loss: 0.2099 - accura
cy: 0.9863 - IOU: 0.6576 - dice_coef: 0.7901 - val_loss: 0.2378 - val_accurac
y: 0.9824 - val_IOU: 0.6211 - val_dice_coef: 0.7622
Epoch 16/100
Learning rate:  0.001
27/27 [==============================] - 186s 7s/step - loss: 0.1964 - accura
cy: 0.9875 - IOU: 0.6769 - dice_coef: 0.8036 - val_loss: 0.5896 - val_accurac
y: 0.9692 - val_IOU: 0.2731 - val_dice_coef: 0.4104
Epoch 17/100
Learning rate:  0.001
27/27 [==============================] - 182s 7s/step - loss: 0.1973 - accura
cy: 0.9860 - IOU: 0.6759 - dice_coef: 0.8027 - val_loss: 0.3991 - val_accurac
y: 0.9754 - val_IOU: 0.4417 - val_dice_coef: 0.6009
Epoch 18/100
Learning rate:  0.001
27/27 [==============================] - 183s 7s/step - loss: 0.1938 - accura
cy: 0.9866 - IOU: 0.6808 - dice_coef: 0.8062 - val_loss: 0.2230 - val_accurac
y: 0.9839 - val_IOU: 0.6431 - val_dice_coef: 0.7770
Epoch 19/100
Learning rate:  0.001
27/27 [==============================] - 184s 7s/step - loss: 0.1902 - accura
cy: 0.9868 - IOU: 0.6853 - dice_coef: 0.8098 - val_loss: 0.3359 - val_accurac
y: 0.9766 - val_IOU: 0.5027 - val_dice_coef: 0.6641
Epoch 20/100
Learning rate:  0.001
27/27 [==============================] - 184s 7s/step - loss: 0.1898 - accura
cy: 0.9865 - IOU: 0.6848 - dice_coef: 0.8102 - val_loss: 0.1842 - val_accurac
y: 0.9856 - val_IOU: 0.6908 - val_dice_coef: 0.8158
Epoch 21/100
Learning rate:  0.001
27/27 [==============================] - 185s 7s/step - loss: 0.1902 - accura
cy: 0.9872 - IOU: 0.6848 - dice_coef: 0.8098 - val_loss: 0.2045 - val_accurac
y: 0.9842 - val_IOU: 0.6628 - val_dice_coef: 0.7955
Epoch 22/100
Learning rate:  0.001
27/27 [==============================] - 184s 7s/step - loss: 0.2214 - accura
cy: 0.9846 - IOU: 0.6420 - dice_coef: 0.7786 - val_loss: 0.1900 - val_accurac
y: 0.9845 - val_IOU: 0.6836 - val_dice_coef: 0.8100
Epoch 23/100
Learning rate:  0.001
27/27 [==============================] - 185s 7s/step - loss: 0.1800 - accura
cy: 0.9873 - IOU: 0.6979 - dice_coef: 0.8200 - val_loss: 0.1661 - val_accurac
y: 0.9867 - val_IOU: 0.7168 - val_dice_coef: 0.8339
Epoch 24/100
Learning rate:  0.001
27/27 [==============================] - 183s 7s/step - loss: 0.2057 - accura
cy: 0.9875 - IOU: 0.6654 - dice_coef: 0.7943 - val_loss: 0.1732 - val_accurac
y: 0.9861 - val_IOU: 0.7063 - val_dice_coef: 0.8268
Epoch 25/100
Learning rate:  0.001
27/27 [==============================] - 183s 7s/step - loss: 0.1627 - accura
cy: 0.9884 - IOU: 0.7237 - dice_coef: 0.8373 - val_loss: 0.1873 - val_accurac
y: 0.9856 - val_IOU: 0.6859 - val_dice_coef: 0.8127
Epoch 26/100
```

```
Learning rate:  0.001
27/27 [==============================] - 185s 7s/step - loss: 0.1820 - accura
cy: 0.9875 - IOU: 0.6962 - dice_coef: 0.8180 - val_loss: 0.1733 - val_accurac
y: 0.9865 - val_IOU: 0.7055 - val_dice_coef: 0.8267
Epoch 27/100
Learning rate:  0.001
27/27 [==============================] - 182s 7s/step - loss: 0.1977 - accura
cy: 0.9872 - IOU: 0.6736 - dice_coef: 0.8023 - val_loss: 0.1665 - val_accurac
y: 0.9865 - val_IOU: 0.7165 - val_dice_coef: 0.8335
Epoch 28/100
Learning rate:  0.001
27/27 [==============================] - 186s 7s/step - loss: 0.1977 - accura
cy: 0.9873 - IOU: 0.6747 - dice_coef: 0.8023 - val_loss: 0.1677 - val_accurac
y: 0.9873 - val_IOU: 0.7139 - val_dice_coef: 0.8323
Epoch 29/100
Learning rate:  0.001
27/27 [==============================] - 186s 7s/step - loss: 0.1773 - accura
cy: 0.9878 - IOU: 0.7018 - dice_coef: 0.8227 - val_loss: 0.1617 - val_accurac
y: 0.9873 - val_IOU: 0.7226 - val_dice_coef: 0.8383
Epoch 30/100
Learning rate:  0.001
27/27 [==============================] - 182s 7s/step - loss: 0.1802 - accura
cy: 0.9886 - IOU: 0.6972 - dice_coef: 0.8198 - val_loss: 0.1697 - val_accurac
y: 0.9872 - val_IOU: 0.7112 - val_dice_coef: 0.8303
Epoch 31/100
Learning rate:  0.001
27/27 [==============================] - 184s 7s/step - loss: 0.1831 - accura
cy: 0.9867 - IOU: 0.6946 - dice_coef: 0.8169 - val_loss: 0.1635 - val_accurac
y: 0.9870 - val_IOU: 0.7200 - val_dice_coef: 0.8365
Epoch 32/100
Learning rate:  0.0005
27/27 [==============================] - 183s 7s/step - loss: 0.1733 - accura
cy: 0.9888 - IOU: 0.7070 - dice_coef: 0.8267 - val_loss: 0.1587 - val_accurac
y: 0.9873 - val_IOU: 0.7276 - val_dice_coef: 0.8413
Epoch 33/100
Learning rate:  0.0005
27/27 [==============================] - 183s 7s/step - loss: 0.1701 - accura
cy: 0.9887 - IOU: 0.7129 - dice_coef: 0.8299 - val_loss: 0.1682 - val_accurac
y: 0.9866 - val_IOU: 0.7142 - val_dice_coef: 0.8318
Epoch 34/100
Learning rate:  0.0005
27/27 [==============================] - 183s 7s/step - loss: 0.1604 - accura
cy: 0.9889 - IOU: 0.7258 - dice_coef: 0.8396 - val_loss: 0.1571 - val_accurac
y: 0.9879 - val_IOU: 0.7290 - val_dice_coef: 0.8429
Epoch 35/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1570 - accura
cy: 0.9898 - IOU: 0.7313 - dice_coef: 0.8430 - val_loss: 0.1642 - val_accurac
y: 0.9868 - val_IOU: 0.7191 - val_dice_coef: 0.8358
Epoch 36/100
Learning rate:  0.0005
27/27 [==============================] - 187s 7s/step - loss: 0.1594 - accura
cy: 0.9882 - IOU: 0.7287 - dice_coef: 0.8406 - val_loss: 0.1554 - val_accurac
y: 0.9878 - val_IOU: 0.7321 - val_dice_coef: 0.8446
Epoch 37/100
Learning rate:  0.0005
27/27 [==============================] - 187s 7s/step - loss: 0.1719 - accura
cy: 0.9879 - IOU: 0.7108 - dice_coef: 0.8281 - val_loss: 0.1572 - val_accurac
y: 0.9877 - val_IOU: 0.7292 - val_dice_coef: 0.8428
Epoch 38/100
Learning rate:  0.0005
27/27 [==============================] - 184s 7s/step - loss: 0.1533 - accura
cy: 0.9893 - IOU: 0.7374 - dice_coef: 0.8467 - val_loss: 0.1528 - val_accurac
y: 0.9878 - val_IOU: 0.7366 - val_dice_coef: 0.8472
```

```
Epoch 39/100
Learning rate:  0.0005
27/27 [==============================] - 184s 7s/step - loss: 0.1649 - accura
cy: 0.9889 - IOU: 0.7201 - dice_coef: 0.8351 - val_loss: 0.1619 - val_accurac
y: 0.9875 - val_IOU: 0.7221 - val_dice_coef: 0.8381
Epoch 40/100
Learning rate:  0.0005
27/27 [==============================] - 185s 7s/step - loss: 0.1769 - accura
cy: 0.9885 - IOU: 0.7027 - dice_coef: 0.8231 - val_loss: 0.1572 - val_accurac
y: 0.9875 - val_IOU: 0.7298 - val_dice_coef: 0.8428
Epoch 41/100
Learning rate:  0.0005
27/27 [==============================] - 187s 7s/step - loss: 0.1518 - accura
cy: 0.9900 - IOU: 0.7398 - dice_coef: 0.8482 - val_loss: 0.1444 - val_accurac
y: 0.9888 - val_IOU: 0.7492 - val_dice_coef: 0.8556
Epoch 42/100
Learning rate:  0.0005
27/27 [==============================] - 187s 7s/step - loss: 0.1680 - accura
cy: 0.9875 - IOU: 0.7150 - dice_coef: 0.8320 - val_loss: 0.1569 - val_accurac
y: 0.9875 - val_IOU: 0.7307 - val_dice_coef: 0.8431
Epoch 43/100
Learning rate:  0.0005
27/27 [==============================] - 189s 7s/step - loss: 0.1623 - accura
cy: 0.9886 - IOU: 0.7247 - dice_coef: 0.8377 - val_loss: 0.1632 - val_accurac
y: 0.9872 - val_IOU: 0.7206 - val_dice_coef: 0.8368
Epoch 44/100
Learning rate:  0.0005
27/27 [==============================] - 290s 11s/step - loss: 0.1462 - accur
acy: 0.9898 - IOU: 0.7472 - dice_coef: 0.8538 - val_loss: 0.1467 - val_accura
cy: 0.9887 - val_IOU: 0.7453 - val_dice_coef: 0.8533
Epoch 45/100
Learning rate:  0.0005
27/27 [==============================] - 215s 8s/step - loss: 0.1630 - accura
cy: 0.9884 - IOU: 0.7229 - dice_coef: 0.8370 - val_loss: 0.1716 - val_accurac
y: 0.9867 - val_IOU: 0.7079 - val_dice_coef: 0.8284
Epoch 46/100
Learning rate:  0.0005
27/27 [==============================] - 193s 7s/step - loss: 0.1672 - accura
cy: 0.9885 - IOU: 0.7173 - dice_coef: 0.8328 - val_loss: 0.1545 - val_accurac
y: 0.9879 - val_IOU: 0.7333 - val_dice_coef: 0.8455
Epoch 47/100
Learning rate:  0.0005
27/27 [==============================] - 184s 7s/step - loss: 0.1635 - accura
cy: 0.9892 - IOU: 0.7219 - dice_coef: 0.8365 - val_loss: 0.1767 - val_accurac
y: 0.9856 - val_IOU: 0.7007 - val_dice_coef: 0.8233
Epoch 48/100
Learning rate:  0.0005
27/27 [==============================] - 193s 7s/step - loss: 0.1618 - accura
cy: 0.9896 - IOU: 0.7242 - dice_coef: 0.8382 - val_loss: 0.1617 - val_accurac
y: 0.9867 - val_IOU: 0.7235 - val_dice_coef: 0.8383
Epoch 49/100
Learning rate:  0.0005
27/27 [==============================] - 191s 7s/step - loss: 0.1704 - accura
cy: 0.9889 - IOU: 0.7122 - dice_coef: 0.8296 - val_loss: 0.1532 - val_accurac
y: 0.9877 - val_IOU: 0.7359 - val_dice_coef: 0.8468
Epoch 50/100
Learning rate:  0.0005
27/27 [==============================] - 187s 7s/step - loss: 0.1742 - accura
cy: 0.9887 - IOU: 0.7064 - dice_coef: 0.8258 - val_loss: 0.1562 - val_accurac
y: 0.9876 - val_IOU: 0.7307 - val_dice_coef: 0.8438
Epoch 51/100
Learning rate:  0.0005
27/27 [==============================] - 189s 7s/step - loss: 0.1657 - accura
cy: 0.9890 - IOU: 0.7185 - dice_coef: 0.8343 - val_loss: 0.1575 - val_accurac
```

```
y: 0.9877 - val_IOU: 0.7291 - val_dice_coef: 0.8425
Epoch 52/100
Learning rate:  0.0005
27/27 [==============================] - 191s 7s/step - loss: 0.1473 - accura
cy: 0.9897 - IOU: 0.7450 - dice_coef: 0.8527 - val_loss: 0.1729 - val_accurac
y: 0.9870 - val_IOU: 0.7062 - val_dice_coef: 0.8271
Epoch 53/100
Learning rate:  0.0005
27/27 [==============================] - 188s 7s/step - loss: 0.1605 - accura
cy: 0.9894 - IOU: 0.7264 - dice_coef: 0.8395 - val_loss: 0.1652 - val_accurac
y: 0.9873 - val_IOU: 0.7180 - val_dice_coef: 0.8348
Epoch 54/100
Learning rate:  0.0005
27/27 [==============================] - 189s 7s/step - loss: 0.1550 - accura
cy: 0.9891 - IOU: 0.7340 - dice_coef: 0.8450 - val_loss: 0.1728 - val_accurac
y: 0.9868 - val_IOU: 0.7076 - val_dice_coef: 0.8272
Epoch 55/100
Learning rate:  0.0005
27/27 [==============================] - 189s 7s/step - loss: 0.1894 - accura
cy: 0.9872 - IOU: 0.6867 - dice_coef: 0.8106 - val_loss: 0.1668 - val_accurac
y: 0.9866 - val_IOU: 0.7172 - val_dice_coef: 0.8332
Epoch 56/100
Learning rate:  0.0005
27/27 [==============================] - 270s 10s/step - loss: 0.1465 - accur
acy: 0.9899 - IOU: 0.7469 - dice_coef: 0.8535 - val_loss: 0.1626 - val_accura
cy: 0.9869 - val_IOU: 0.7226 - val_dice_coef: 0.8374
Epoch 57/100
Learning rate:  0.0005
27/27 [==============================] - 200s 7s/step - loss: 0.1522 - accura
cy: 0.9895 - IOU: 0.7381 - dice_coef: 0.8478 - val_loss: 0.1586 - val_accurac
y: 0.9872 - val_IOU: 0.7280 - val_dice_coef: 0.8414
Epoch 58/100
Learning rate:  0.0005
27/27 [==============================] - 213s 8s/step - loss: 0.1570 - accura
cy: 0.9895 - IOU: 0.7304 - dice_coef: 0.8430 - val_loss: 0.1727 - val_accurac
y: 0.9855 - val_IOU: 0.7077 - val_dice_coef: 0.8273
Epoch 59/100
Learning rate:  0.0005
27/27 [==============================] - 187s 7s/step - loss: 0.1616 - accura
cy: 0.9884 - IOU: 0.7249 - dice_coef: 0.8384 - val_loss: 0.1565 - val_accurac
y: 0.9872 - val_IOU: 0.7309 - val_dice_coef: 0.8435
Epoch 60/100
Learning rate:  0.0005
27/27 [==============================] - 181s 7s/step - loss: 0.1664 - accura
cy: 0.9900 - IOU: 0.7181 - dice_coef: 0.8336 - val_loss: 0.1505 - val_accurac
y: 0.9884 - val_IOU: 0.7394 - val_dice_coef: 0.8495
Epoch 61/100
Learning rate:  0.0005
27/27 [==============================] - 190s 7s/step - loss: 0.1734 - accura
cy: 0.9878 - IOU: 0.7071 - dice_coef: 0.8266 - val_loss: 0.1651 - val_accurac
y: 0.9865 - val_IOU: 0.7192 - val_dice_coef: 0.8349
Epoch 62/100
Learning rate:  0.0005
27/27 [==============================] - 279s 10s/step - loss: 0.1509 - accur
acy: 0.9893 - IOU: 0.7408 - dice_coef: 0.8491 - val_loss: 0.1551 - val_accura
cy: 0.9881 - val_IOU: 0.7325 - val_dice_coef: 0.8449
Epoch 63/100
Learning rate:  0.0005
27/27 [==============================] - 281s 11s/step - loss: 0.1593 - accur
acy: 0.9897 - IOU: 0.7277 - dice_coef: 0.8407 - val_loss: 0.1519 - val_accura
cy: 0.9876 - val_IOU: 0.7383 - val_dice_coef: 0.8481
Epoch 64/100
Learning rate:  0.0005
27/27 [==============================] - 188s 7s/step - loss: 0.1577 - accura
```

```
cy: 0.9889 - IOU: 0.7301 - dice_coef: 0.8423 - val_loss: 0.1428 - val_accurac
y: 0.9889 - val_IOU: 0.7513 - val_dice_coef: 0.8572
Epoch 65/100
Learning rate:  0.0005
27/27 [==============================] - 189s 7s/step - loss: 0.1614 - accura
cy: 0.9888 - IOU: 0.7277 - dice_coef: 0.8386 - val_loss: 0.2975 - val_accurac
y: 0.9791 - val_IOU: 0.5465 - val_dice_coef: 0.7025
Epoch 66/100
Learning rate:  0.0005
27/27 [==============================] - 197s 7s/step - loss: 0.1621 - accura
cy: 0.9891 - IOU: 0.7236 - dice_coef: 0.8379 - val_loss: 0.1576 - val_accurac
y: 0.9876 - val_IOU: 0.7291 - val_dice_coef: 0.8424
Epoch 67/100
Learning rate:  0.0005
27/27 [==============================] - 196s 7s/step - loss: 0.1547 - accura
cy: 0.9894 - IOU: 0.7354 - dice_coef: 0.8453 - val_loss: 0.1566 - val_accurac
y: 0.9871 - val_IOU: 0.7308 - val_dice_coef: 0.8434
Epoch 68/100
Learning rate:  0.0005
27/27 [==============================] - 188s 7s/step - loss: 0.1550 - accura
cy: 0.9895 - IOU: 0.7341 - dice_coef: 0.8450 - val_loss: 0.1518 - val_accurac
y: 0.9881 - val_IOU: 0.7381 - val_dice_coef: 0.8482
Epoch 69/100
Learning rate:  0.0005
27/27 [==============================] - 189s 7s/step - loss: 0.1546 - accura
cy: 0.9894 - IOU: 0.7346 - dice_coef: 0.8454 - val_loss: 0.1630 - val_accurac
y: 0.9872 - val_IOU: 0.7215 - val_dice_coef: 0.8370
Epoch 70/100
Learning rate:  0.0005
27/27 [==============================] - 188s 7s/step - loss: 0.1543 - accura
cy: 0.9892 - IOU: 0.7347 - dice_coef: 0.8457 - val_loss: 0.1580 - val_accurac
y: 0.9872 - val_IOU: 0.7282 - val_dice_coef: 0.8420
Epoch 71/100
Learning rate:  0.0005
27/27 [==============================] - 187s 7s/step - loss: 0.1814 - accura
cy: 0.9882 - IOU: 0.6982 - dice_coef: 0.8186 - val_loss: 0.1538 - val_accurac
y: 0.9880 - val_IOU: 0.7342 - val_dice_coef: 0.8462
Epoch 72/100
Learning rate:  0.0005
27/27 [==============================] - 190s 7s/step - loss: 0.1527 - accura
cy: 0.9893 - IOU: 0.7375 - dice_coef: 0.8473 - val_loss: 0.1537 - val_accurac
y: 0.9877 - val_IOU: 0.7350 - val_dice_coef: 0.8463
Epoch 73/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1481 - accura
cy: 0.9896 - IOU: 0.7443 - dice_coef: 0.8519 - val_loss: 0.1545 - val_accurac
y: 0.9880 - val_IOU: 0.7339 - val_dice_coef: 0.8455
Epoch 74/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1546 - accura
cy: 0.9894 - IOU: 0.7356 - dice_coef: 0.8454 - val_loss: 0.1456 - val_accurac
y: 0.9886 - val_IOU: 0.7467 - val_dice_coef: 0.8544
Epoch 75/100
Learning rate:  0.0005
27/27 [==============================] - 186s 7s/step - loss: 0.1596 - accura
cy: 0.9898 - IOU: 0.7278 - dice_coef: 0.8404 - val_loss: 0.1542 - val_accurac
y: 0.9881 - val_IOU: 0.7340 - val_dice_coef: 0.8458
Epoch 76/100
Learning rate:  0.0005
27/27 [==============================] - 190s 7s/step - loss: 0.1589 - accura
cy: 0.9887 - IOU: 0.7287 - dice_coef: 0.8411 - val_loss: 0.1583 - val_accurac
y: 0.9876 - val_IOU: 0.7283 - val_dice_coef: 0.8417
Epoch 77/100
Learning rate:  0.0005
```

```
27/27 [==============================] - 189s 7s/step - loss: 0.1479 - accura
cy: 0.9903 - IOU: 0.7445 - dice_coef: 0.8521 - val_loss: 0.1556 - val_accurac
y: 0.9879 - val_IOU: 0.7322 - val_dice_coef: 0.8444
Epoch 78/100
Learning rate:  0.0005
27/27 [==============================] - 193s 7s/step - loss: 0.1549 - accura
cy: 0.9891 - IOU: 0.7340 - dice_coef: 0.8451 - val_loss: 0.1484 - val_accurac
y: 0.9883 - val_IOU: 0.7429 - val_dice_coef: 0.8516
Epoch 79/100
Learning rate:  0.0005
27/27 [==============================] - 197s 7s/step - loss: 0.1493 - accura
cy: 0.9896 - IOU: 0.7423 - dice_coef: 0.8507 - val_loss: 0.1599 - val_accurac
y: 0.9871 - val_IOU: 0.7264 - val_dice_coef: 0.8401
Epoch 80/100
Learning rate:  0.0005
27/27 [==============================] - 194s 7s/step - loss: 0.1580 - accura
cy: 0.9899 - IOU: 0.7291 - dice_coef: 0.8420 - val_loss: 0.1494 - val_accurac
y: 0.9881 - val_IOU: 0.7418 - val_dice_coef: 0.8506
Epoch 81/100
Learning rate:  0.0005
27/27 [==============================] - 194s 7s/step - loss: 0.1451 - accura
cy: 0.9898 - IOU: 0.7485 - dice_coef: 0.8549 - val_loss: 0.1506 - val_accurac
y: 0.9884 - val_IOU: 0.7400 - val_dice_coef: 0.8494
Epoch 82/100
Learning rate:  0.0005
27/27 [==============================] - 191s 7s/step - loss: 0.1576 - accura
cy: 0.9893 - IOU: 0.7304 - dice_coef: 0.8424 - val_loss: 0.1461 - val_accurac
y: 0.9889 - val_IOU: 0.7463 - val_dice_coef: 0.8539
Epoch 83/100
Learning rate:  0.0005
27/27 [==============================] - 181s 7s/step - loss: 0.1563 - accura
cy: 0.9897 - IOU: 0.7332 - dice_coef: 0.8437 - val_loss: 0.1509 - val_accurac
y: 0.9886 - val_IOU: 0.7400 - val_dice_coef: 0.8491
Epoch 84/100
Learning rate:  0.0005
27/27 [==============================] - 179s 7s/step - loss: 0.1398 - accura
cy: 0.9907 - IOU: 0.7563 - dice_coef: 0.8602 - val_loss: 0.1532 - val_accurac
y: 0.9887 - val_IOU: 0.7358 - val_dice_coef: 0.8468
Epoch 85/100
Learning rate:  0.0005
27/27 [==============================] - 181s 7s/step - loss: 0.1554 - accura
cy: 0.9899 - IOU: 0.7330 - dice_coef: 0.8446 - val_loss: 0.1526 - val_accurac
y: 0.9879 - val_IOU: 0.7364 - val_dice_coef: 0.8474
Epoch 86/100
Learning rate:  0.0005
27/27 [==============================] - 181s 7s/step - loss: 0.1440 - accura
cy: 0.9901 - IOU: 0.7512 - dice_coef: 0.8560 - val_loss: 0.1416 - val_accurac
y: 0.9888 - val_IOU: 0.7533 - val_dice_coef: 0.8584
Epoch 87/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1561 - accura
cy: 0.9896 - IOU: 0.7318 - dice_coef: 0.8439 - val_loss: 0.1474 - val_accurac
y: 0.9885 - val_IOU: 0.7444 - val_dice_coef: 0.8526
Epoch 88/100
Learning rate:  0.0005
27/27 [==============================] - 180s 7s/step - loss: 0.1577 - accura
cy: 0.9889 - IOU: 0.7309 - dice_coef: 0.8423 - val_loss: 0.1710 - val_accurac
y: 0.9864 - val_IOU: 0.7098 - val_dice_coef: 0.8290
Epoch 89/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1579 - accura
cy: 0.9897 - IOU: 0.7303 - dice_coef: 0.8421 - val_loss: 0.1472 - val_accurac
y: 0.9887 - val_IOU: 0.7448 - val_dice_coef: 0.8528
Epoch 90/100
```

```
Learning rate:  0.0005
27/27 [==============================] - 183s 7s/step - loss: 0.1520 - accura
cy: 0.9896 - IOU: 0.7377 - dice_coef: 0.8480 - val_loss: 0.1738 - val_accurac
y: 0.9874 - val_IOU: 0.7070 - val_dice_coef: 0.8262
Epoch 91/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1519 - accura
cy: 0.9898 - IOU: 0.7389 - dice_coef: 0.8481 - val_loss: 0.1472 - val_accurac
y: 0.9886 - val_IOU: 0.7447 - val_dice_coef: 0.8528
Epoch 92/100
Learning rate:  0.0005
27/27 [==============================] - 183s 7s/step - loss: 0.1569 - accura
cy: 0.9896 - IOU: 0.7307 - dice_coef: 0.8431 - val_loss: 0.1567 - val_accurac
y: 0.9873 - val_IOU: 0.7302 - val_dice_coef: 0.8433
Epoch 93/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1620 - accura
cy: 0.9891 - IOU: 0.7248 - dice_coef: 0.8380 - val_loss: 0.1479 - val_accurac
y: 0.9883 - val_IOU: 0.7439 - val_dice_coef: 0.8521
Epoch 94/100
Learning rate:  0.0005
27/27 [==============================] - 179s 7s/step - loss: 0.1597 - accura
cy: 0.9897 - IOU: 0.7273 - dice_coef: 0.8403 - val_loss: 0.1381 - val_accurac
y: 0.9893 - val_IOU: 0.7586 - val_dice_coef: 0.8619
Epoch 95/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1522 - accura
cy: 0.9892 - IOU: 0.7386 - dice_coef: 0.8478 - val_loss: 0.1485 - val_accurac
y: 0.9883 - val_IOU: 0.7435 - val_dice_coef: 0.8515
Epoch 96/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1487 - accura
cy: 0.9898 - IOU: 0.7440 - dice_coef: 0.8513 - val_loss: 0.1535 - val_accurac
y: 0.9879 - val_IOU: 0.7348 - val_dice_coef: 0.8465
Epoch 97/100
Learning rate:  0.0005
27/27 [==============================] - 183s 7s/step - loss: 0.1525 - accura
cy: 0.9893 - IOU: 0.7377 - dice_coef: 0.8475 - val_loss: 0.1708 - val_accurac
y: 0.9866 - val_IOU: 0.7091 - val_dice_coef: 0.8292
Epoch 98/100
Learning rate:  0.0005
27/27 [==============================] - 182s 7s/step - loss: 0.1495 - accura
cy: 0.9892 - IOU: 0.7430 - dice_coef: 0.8505 - val_loss: 0.1603 - val_accurac
y: 0.9877 - val_IOU: 0.7249 - val_dice_coef: 0.8397
Epoch 99/100
Learning rate:  0.0005
27/27 [==============================] - 184s 7s/step - loss: 0.1500 - accura
cy: 0.9906 - IOU: 0.7410 - dice_coef: 0.8500 - val_loss: 0.1925 - val_accurac
y: 0.9860 - val_IOU: 0.6779 - val_dice_coef: 0.8075
Epoch 100/100
Learning rate:  0.0005
27/27 [==============================] - 181s 7s/step - loss: 0.1489 - accura
cy: 0.9899 - IOU: 0.7440 - dice_coef: 0.8511 - val_loss: 0.1775 - val_accurac
y: 0.9868 - val_IOU: 0.6987 - val_dice_coef: 0.8225
```

In [70]:

```
# save model
model.save('deepcrack.h5')
print('Model Saved!')
```

```
Model Saved!
```

In [71]:

```
# saving and loading the model weights
```

```
# save model
model.save_weights('deepcrack_weights')
print('Model Saved!')

# load model
# savedModel = model.load_weights('gfgModelWeights')
# print('Model Loaded!')
```

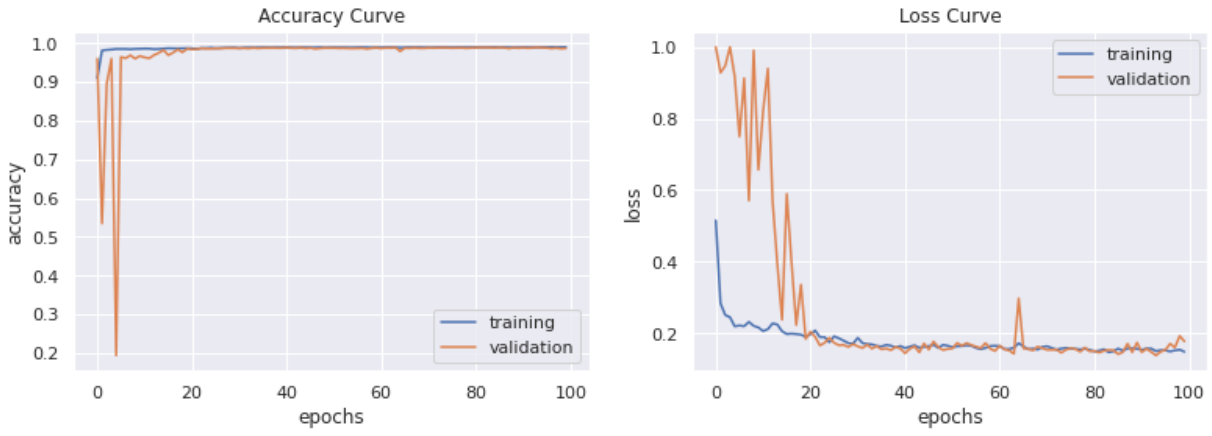
```
Model Saved!
```

In [72]:

```
train_loss = history.history['loss']
valid_loss = history.history['val_loss']

train_acc = history.history['accuracy']
valid_acc = history.history['val_accuracy']
```

In [73]:

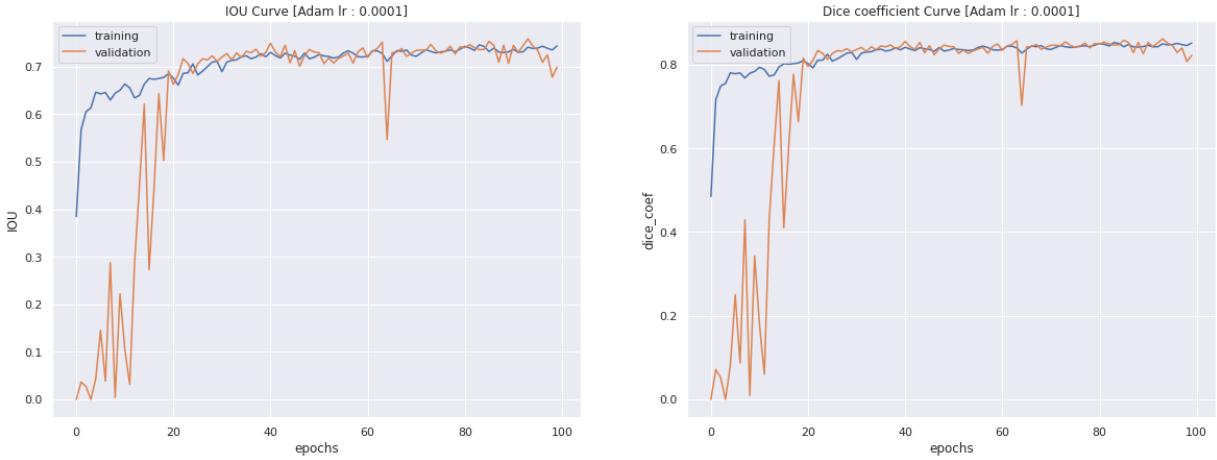
```
fig, axes = plt.subplots(1, 2, figsize=(13,4))
axes = axes.flatten()


axes[0].plot(train_acc, label='training')
axes[0].plot(valid_acc, label='validation')
axes[0].set_title('Accuracy Curve')
axes[0].set_xlabel('epochs')
axes[0].set_ylabel('accuracy')
axes[0].legend()



axes[1].plot(train_loss, label='training')
axes[1].plot(valid_loss, label='validation')
axes[1].set_title('Loss Curve')
axes[1].set_xlabel('epochs')
axes[1].set_ylabel('loss')
axes[1].legend()

plt.show()
```



In [74]:

```
train_dice = history.history['dice_coef']
valid_dice = history.history['val_dice_coef']


train_IOU = history.history['IOU']
valid_IOU = history.history['val_IOU']
```

In [75]:
```
fig, axes = plt.subplots(1, 2, figsize=(20,7))
axes = axes.flatten()

axes[0].plot(train_IOU, label='training')
axes[0].plot(valid_IOU, label='validation')
axes[0].set_title('IOU Curve [Adam lr : 0.0001]')
axes[0].set_xlabel('epochs')
axes[0].set_ylabel('IOU')
axes[0].legend()


axes[1].plot(train_dice, label='training')
axes[1].plot(valid_dice, label='validation')
axes[1].set_title('Dice coefficient Curve [Adam lr : 0.0001]')
axes[1].set_xlabel('epochs')
axes[1].set_ylabel('dice_coef')
axes[1].legend()

plt.show()
```



Testing

In [76]:
```
test_generator = Generator(test_image_paths, test_mask_paths,237, img_dim)

for x_test, y_test in test_generator:
  break

y_pred = model.predict(x_test)

yy_true = (y_test>0.5).flatten()
yy_pred = (y_pred>0.5).flatten()
```

In [77]:
```
report = classification_report(yy_true, yy_pred, output_dict=True)

Accuracy = accuracy_score(yy_true, yy_pred)

Precision = report['True']['precision']
Recall = report['True']['recall']
F1_score = report['True']['f1-score']

Sensitivity = Recall
Specificity = report['False']['recall']

AUC = roc_auc_score(y_test.flatten(), y_pred.flatten())
```

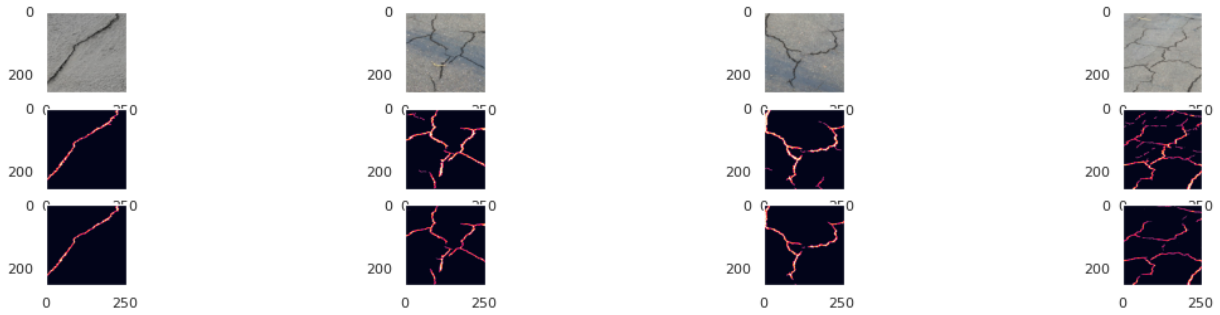
```
IOU = (Precision*Recall)/(Precision+Recall-Precision*Recall)

print("Accuracy: {0:.4f}\n".format(Accuracy))
print("Precision: {0:.4f}\n".format(Precision))
print("Recall: {0:.4f}\n".format(Recall))
print("F1-Score: {0:.4f}\n".format(F1_score))
print("Sensitivity: {0:.4f}\n".format(Sensitivity))
print("Specificity: {0:.4f}\n".format(Specificity))
print("AUC: {0:.4f}\n".format(AUC))
print("IOU: {0:.4f}\n".format(IOU))
print('-'*50,'\n')
print(classification_report(yy_true, yy_pred))
```

```
Accuracy: 0.9841

Precision: 0.9504

Recall: 0.7080

F1-Score: 0.8115

Sensitivity: 0.7080

Specificity: 0.9981

AUC: 0.9009

IOU: 0.6827

--------------------------------------------------

              precision    recall  f1-score   support

       False       0.99      1.00      0.99  14780073
        True       0.95      0.71      0.81    751959

    accuracy                           0.98  15532032
   macro avg       0.97      0.85      0.90  15532032
weighted avg       0.98      0.98      0.98  15532032
```

In [80]:

```
test_generator = Generator(test_image_paths, test_mask_paths,237, img_dim)

for x_test, y_test in test_generator:
  break

y_pred = model.predict(x_test)

plt.figure(figsize=(40, 4))
for i in range(4):
    # display image
    plt.grid(False)
    ax = plt.subplot(3, 8, i + 1)
    plt.imshow(x_test[i])
    # display original label
    plt.grid(False)
    ax = plt.subplot(3, 8, 8+i + 1)
    plt.imshow(y_test[i])

    # display reconstructed (after noise removed) image
    plt.grid(False)
    ax = plt.subplot(3, 8, 16 +i+ 1)
    plt.imshow(y_pred[i])
```

```
plt.grid(False)
plt.show()
```



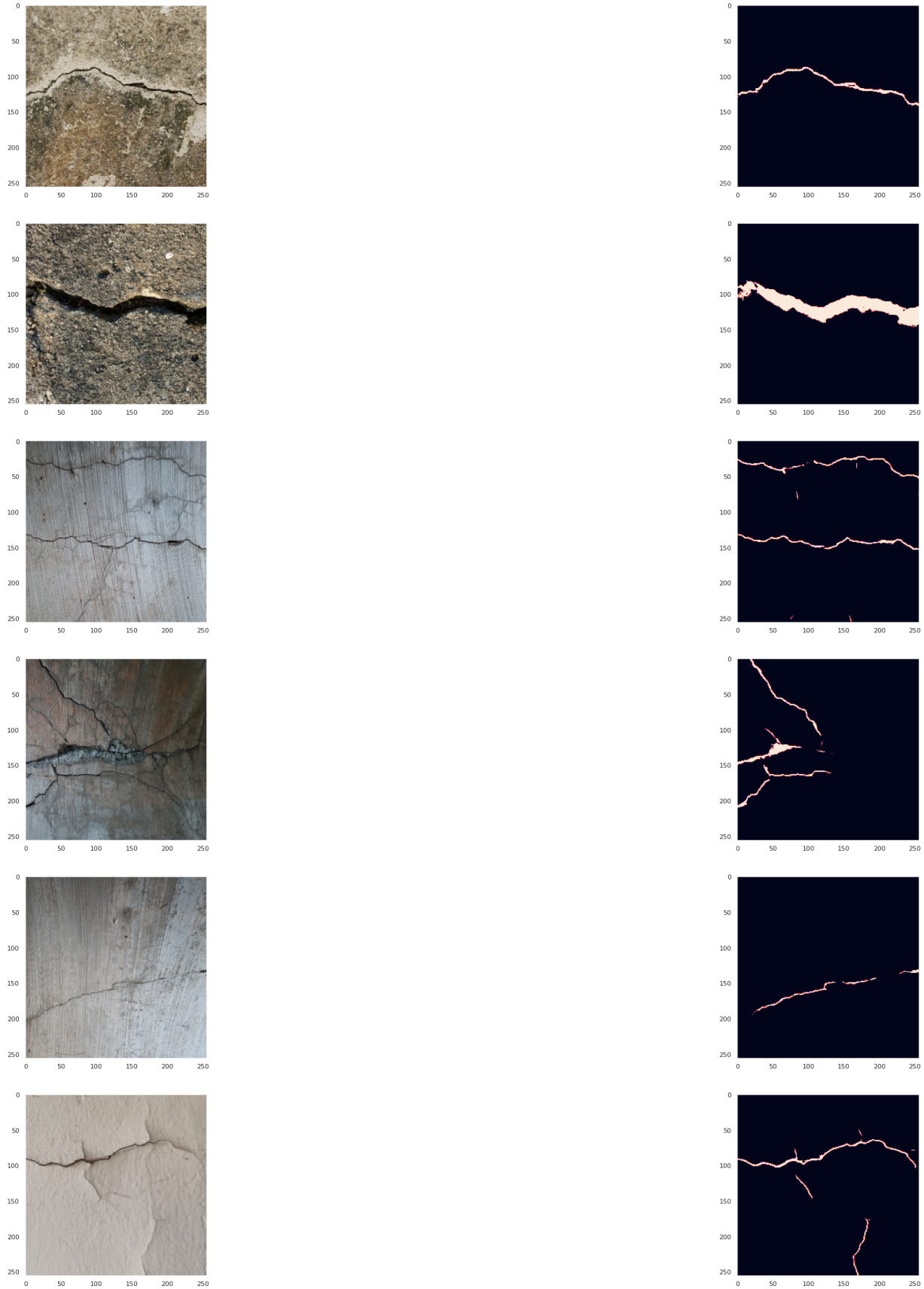
In [81]:

```
print(x_test.shape)
print(y_test.shape)
print(y_pred.shape)
```

```
(237, 256, 256, 3)
(237, 256, 256, 1)
(237, 256, 256, 1)
```

In [82]:

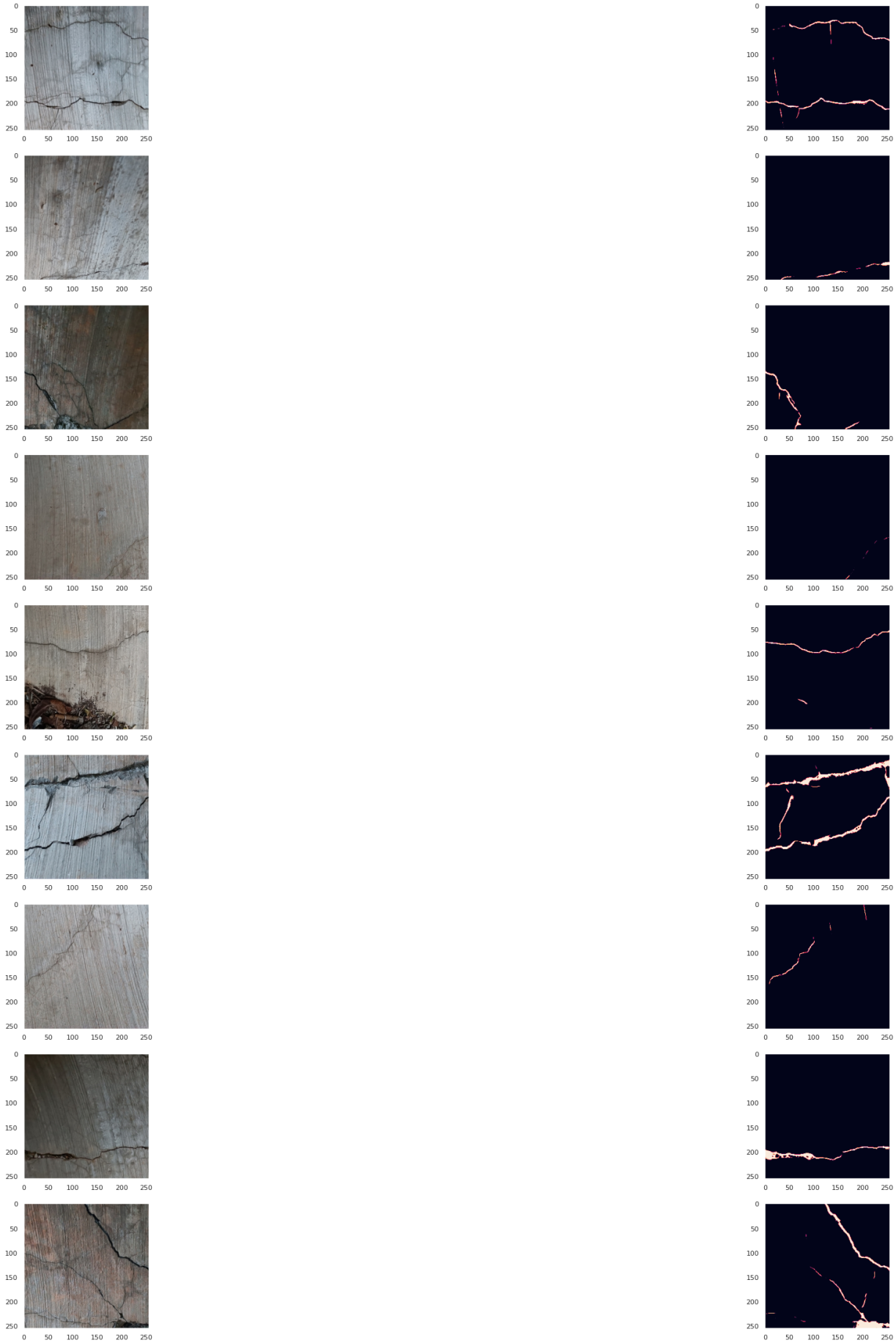
```
# campus_image_dir = r'/home/ubuntu/Desktop/NNDL Project/campus_crack'
path = r'/home/ubuntu/Downloads/pipe_images_256'
pipe_image_paths = sorted([os.path.join(path, fname) for fname in os.listdir(

pipe = Generator(pipe_image_paths, pipe_image_paths, batch_size, img_dim, aug
for p_test, y_test in pipe:
    break
#print(images.shape)

pipe_pred = model.predict(p_test)

n=len(p_test)

plt.figure(figsize=(40, 40))
for i in range(n):
    # display image
    plt.grid(False)
    plt.subplot(n,2,2*i+1)
    plt.imshow(p_test[i])

    # display reconstructed (after noise removed) image
    plt.grid(False)
    plt.subplot(n, 2,  2*(i+1))
    plt.imshow(pipe_pred[i])

plt.grid(False)
plt.show()
```



In [83]:

```
path = r'/home/ubuntu/Desktop/NNDL Project/PIPE NEW 256'
pipe_image_paths = sorted([os.path.join(path, fname) for fname in os.listdir(

pipe = Generator(pipe_image_paths, pipe_image_paths, batch_size, img_dim, aug
for p_test, y_test in pipe:
    break
#print(images.shape)

pipe_pred = model.predict(p_test)
```

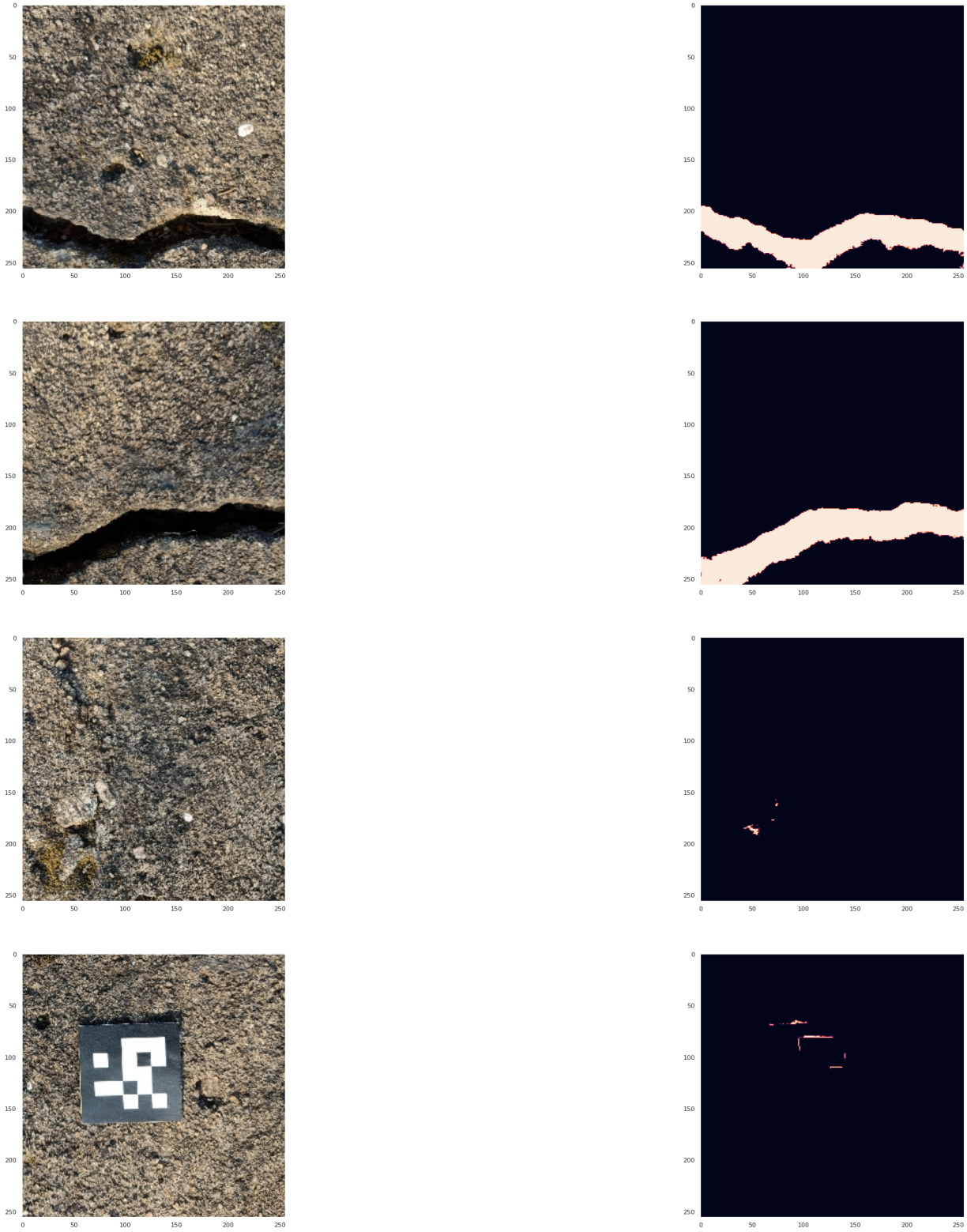
```
n=len(p_test)
# plt.subplot(9,3,1)
# plt.imshow(p_test[3])
# plt.subplot(9,3,2)
# plt.imshow(pipe_pred[3])
plt.figure(figsize=(40, 40))
for i in range(n):
    # display image
    plt.grid(False)
    plt.subplot(n,2,2*i+1)
    plt.imshow(p_test[i])
#     # display original label
#     ax = plt.subplot(3, 8, 8+i + 1)
#     plt.imshow(y_test[i])

    # display reconstructed (after noise removed) image
    plt.grid(False)
    plt.subplot(n, 2,  2*(i+1))
    plt.imshow(pipe_pred[i])

plt.grid(False)
plt.show()
```



In [84]:

```
path = r'/home/ubuntu/Desktop/NNDL Project/PAVEMENT NEW 256'
pipe_image_paths = sorted([os.path.join(path, fname) for fname in os.listdir(

pipe = Generator(pipe_image_paths, pipe_image_paths, batch_size, img_dim, aug
for p_test, y_test in pipe:
    break
```

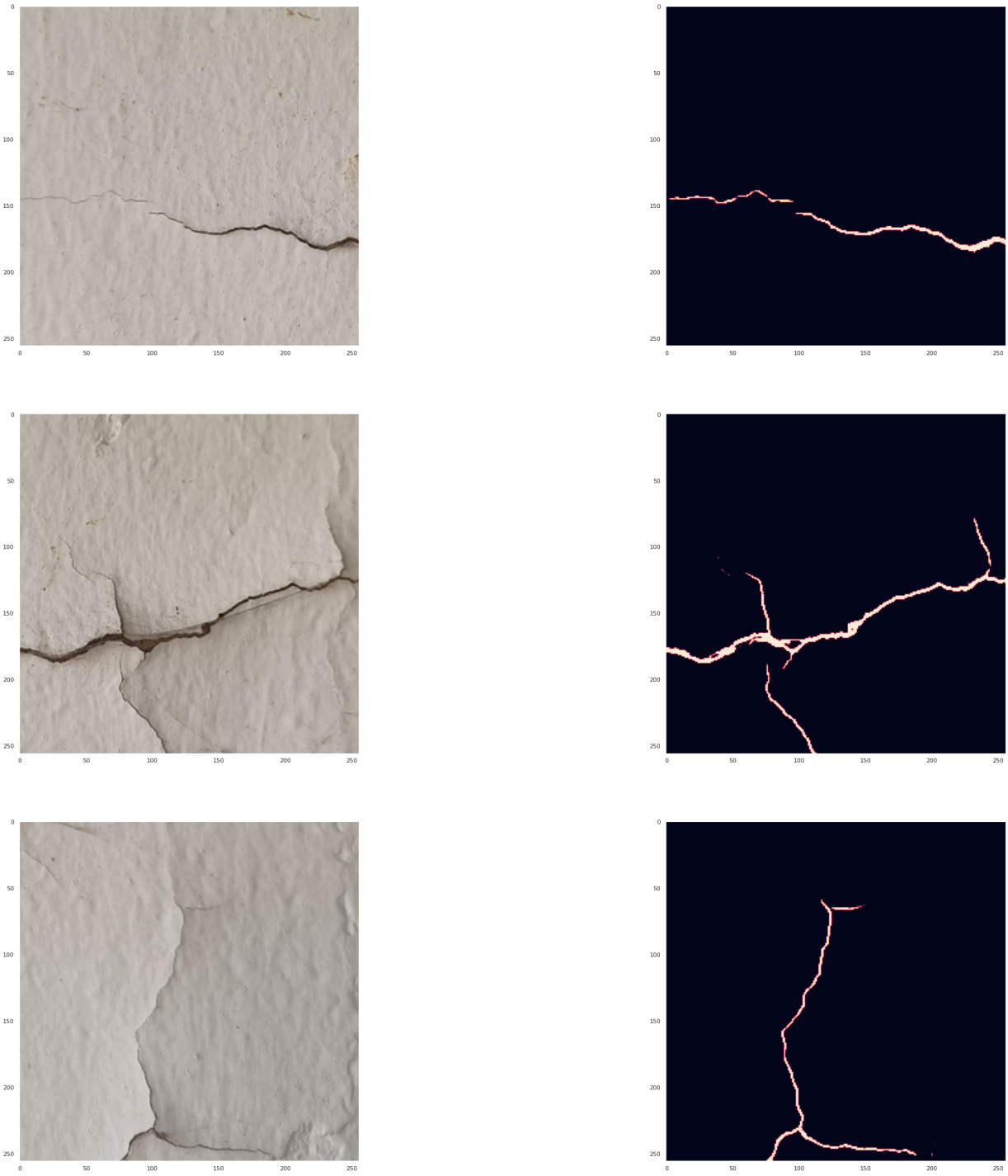
```
#print(images.shape)

pipe_pred = model.predict(p_test)

n=len(p_test)
# plt.subplot(9,3,1)
# plt.imshow(p_test[3])
# plt.subplot(9,3,2)
# plt.imshow(pipe_pred[3])
plt.figure(figsize=(40, 40))
for i in range(n):
    # display image
    plt.grid(False)
    plt.subplot(n,2,2*i+1)
    plt.imshow(p_test[i])
#     # display original label
#     ax = plt.subplot(3, 8, 8+i + 1)
#     plt.imshow(y_test[i])

    # display reconstructed (after noise removed) image
    plt.grid(False)
    plt.subplot(n, 2,  2*(i+1))
    plt.imshow(pipe_pred[i])

plt.grid(False)
plt.show()
```



In [85]:

```
path = r'/home/ubuntu/Desktop/NNDL Project/WALL NEW 256'
pipe_image_paths = sorted([os.path.join(path, fname) for fname in os.listdir(

pipe = Generator(pipe_image_paths, pipe_image_paths, batch_size, img_dim, aug
for p_test, y_test in pipe:
    break
#print(images.shape)

pipe_pred = model.predict(p_test)

n=len(p_test)
# plt.subplot(9,3,1)
# plt.imshow(p_test[3])
# plt.subplot(9,3,2)
```

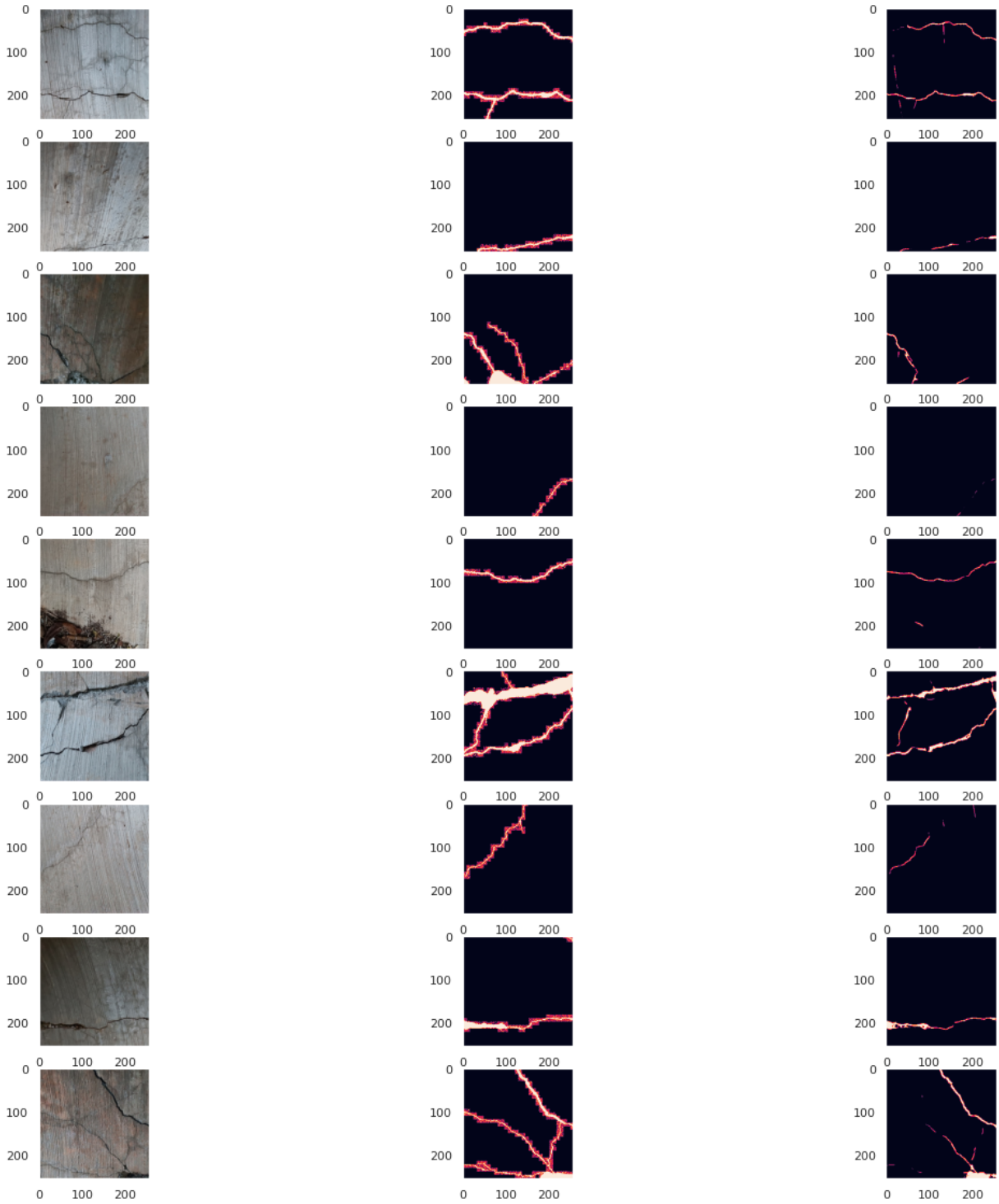
```
# plt.imshow(pipe_pred[3])
plt.figure(figsize=(40, 40))
for i in range(n):
    # display image
    plt.grid(False)
    plt.subplot(n,2,2*i+1)
    plt.imshow(p_test[i])
#     # display original label
#     ax = plt.subplot(3, 8, 8+i + 1)
#     plt.imshow(y_test[i])

    # display reconstructed (after noise removed) image
    plt.grid(False)
    plt.subplot(n, 2,  2*(i+1))
    plt.imshow(pipe_pred[i])

plt.grid(False)
plt.show()
```



In [86]:
```
path = r'/home/ubuntu/Desktop/NNDL Project/PIPE NEW 256'
pipe_image_paths = sorted([os.path.join(path, fname) for fname in os.listdir(

test_lab_path = r'/home/ubuntu/Desktop/NNDL Project/Pipe GT'
pipe_lab_paths = sorted([os.path.join(test_lab_path, fname) for fname in os.l


pipe = Generator(pipe_image_paths, pipe_lab_paths, 9, img_dim, augment = Fals
for p_test, y_test in pipe:
    break
print(p_test.shape)
print(y_test.shape)
y_test = y_test.reshape(y_test.shape[0],y_test.shape[1],y_test.shape[2],y_tes
print(y_test.shape)
y_test = np.expand_dims(y_test[:,:,:,0], axis = -1)
print(y_test.shape)

y_pred = model.predict(p_test)

yy_true = (y_test>0.5).flatten()
yy_pred = (y_pred>0.5).flatten()

print(yy_true.shape)
print(yy_pred.shape)

print((y_test.flatten().shape))
print((y_pred.flatten().shape))

report = classification_report(yy_true, yy_pred, output_dict=True)

Accuracy = accuracy_score(yy_true, yy_pred)

Precision = report['True']['precision']
Recall = report['True']['recall']
F1_score = report['True']['f1-score']

Sensitivity = Recall
Specificity = report['False']['recall']

AUC = roc_auc_score(y_test.flatten(), y_pred.flatten())

IOU = (Precision*Recall)/(Precision+Recall-Precision*Recall)

print("Accuracy: {0:.4f}\n".format(Accuracy))
print("Precision: {0:.4f}\n".format(Precision))
print("Recall: {0:.4f}\n".format(Recall))
print("F1-Score: {0:.4f}\n".format(F1_score))
print("Sensitivity: {0:.4f}\n".format(Sensitivity))
print("Specificity: {0:.4f}\n".format(Specificity))
print("AUC: {0:.4f}\n".format(AUC))
print("IOU: {0:.4f}\n".format(IOU))
print('-'*50,'\n')
print(classification_report(yy_true, yy_pred))

n=len(p_test)

plt.figure(figsize=(20, 20))

for i in range(n):
    # display image
    plt.grid(False)
    plt.subplot(n,3,3*i+1)
    plt.imshow(p_test[i])
```

```
    plt.grid(False)
    plt.subplot(n,3,3*i+2)
    plt.imshow(y_test[i])

    # display reconstructed (after noise removed) image
    plt.grid(False)
    plt.subplot(n, 3,  3*(i+1))
    plt.imshow(y_pred[i])

plt.grid(False)
plt.show()
```

```
(9, 256, 256, 3)
(9, 256, 256, 3, 1)
(9, 256, 256, 3)
(9, 256, 256, 1)
(589824,)
(589824,)
(589824,)
(589824,)
Accuracy: 0.9474

Precision: 0.9472

Recall: 0.2386

F1-Score: 0.3812

Sensitivity: 0.2386

Specificity: 0.9990

AUC: 0.4934

IOU: 0.2355

--------------------------------------------------

              precision    recall  f1-score   support

       False       0.95      1.00      0.97    549788
        True       0.95      0.24      0.38     40036

    accuracy                           0.95    589824
   macro avg       0.95      0.62      0.68    589824
weighted avg       0.95      0.95      0.93    589824
```



In [87]:

```
# # path = r'/home/ubuntu/Desktop/NNDL Project/WALL NEW 256'
# # path = r'/home/ubuntu/Desktop/NNDL Project/PAVEMENT NEW 256'
# path = r'/home/ubuntu/Desktop/NNDL Project/PIPE NEW 256'
# pipe_image_paths = sorted([os.path.join(path, fname) for fname in os.listdi

# pipe = Generator(pipe_image_paths, pipe_image_paths, batch_size, img_dim, a
# for p_test, y_test in pipe:
#     break
# #print(images.shape)

# pipe_pred = model.predict(p_test)

# n=len(p_test)
# # plt.subplot(9,3,1)
# # plt.imshow(p_test[3])
# # plt.subplot(9,3,2)
# plt.imshow(pipe_pred[8])
```

```
# # plt.figure(figsize=(40, 40))
# # for i in range(n):
# #     # display image
# #     plt.subplot(n,2,2*i+1)
# #     plt.imshow(p_test[i])
# # #     # display original label
# # #     ax = plt.subplot(3, 8, 8+i + 1)
# # #     plt.imshow(y_test[i])

# #     # display reconstructed (after noise removed) image
# #     plt.subplot(n, 2,  2*(i+1))
# #     plt.imshow(pipe_pred[i])

# plt.show()
```

In [ ]: