

Hyper Spectral Image Segmentation using FPN

Overview

This is continuation of HSI segmentation case study presented in link :

<https://sachinbu.medium.com/hyperspectral-image-segmentation-21432965e138>

In the study simple neural network was used to classify each pixel in the Hyper Spectral Image.

As mentioned in the above article(section- Alternative Approach), we will consider Convolutional Neural Network (CNN) for HSI segmentation.

U-Net is the CNN model considered for the study. Here two types of model are trained:

1. Pretrained U-Net which has resnet as backbone for encoder section. Convolution layers are added before the pretrained Network to get a 3 channel image which will be fed to the pretrained Network.
2. Simple U-Net trained from scratch.

Same data mentioned in the above article is considered in this study.

To train the above mentioned models, Indian Pines image (145x145x200) is augmented to get 1000 images where 800 images are used for training the model and 200 images are used for validation. Details of generating the images and training the model are captured in this notebook

In [1]:

```
!pip install patchify
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: patchify in /usr/local/lib/python3.7/dist-packages (0.2.3)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.7/dist-packages (from patchify) (1
.21.6)
```

In [2]:

```
import numpy as np
import scipy.io
import matplotlib.pyplot as plt
import patchify as patch
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
import os, time
from datetime import datetime
from scipy.ndimage import rotate
```

Data

In [3]:

```
# Data Source : http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Indian_Pines
!wget wget --header="Host: www.ehu.eus" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes" "http://www.ehu.eus/ccwintco/uploads/6/67/Indian_pines_corrected.mat" -c -O 'Indian_pines_corrected.mat'
```

```
--2022-05-27 02:16:52--  http://wget/
Resolving wget (wget)... failed: Name or service not known.
wget: unable to resolve host address 'wget'
--2022-05-27 02:16:52--  http://www.ehu.eus/ccwintco/uploads/6/67/Indian_pines_corrected.mat
Resolving www.ehu.eus (www.ehu.eus)... 158.227.0.65, 2001:720:1410::65
Connecting to www.ehu.eus (www.ehu.eus)|158.227.0.65|:80... connected.
```

```
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.ehu.eus/ccwintco/uploads/6/67/Indian_pines_corrected.mat [following]
--2022-05-27 02:16:52-- https://www.ehu.eus/ccwintco/uploads/6/67/Indian_pines_corrected.mat
Connecting to www.ehu.eus (www.ehu.eus)|158.227.0.65|:443... connected.
HTTP request sent, awaiting response... 416 Requested Range Not Satisfiable
```

The file is already fully retrieved; nothing to do.

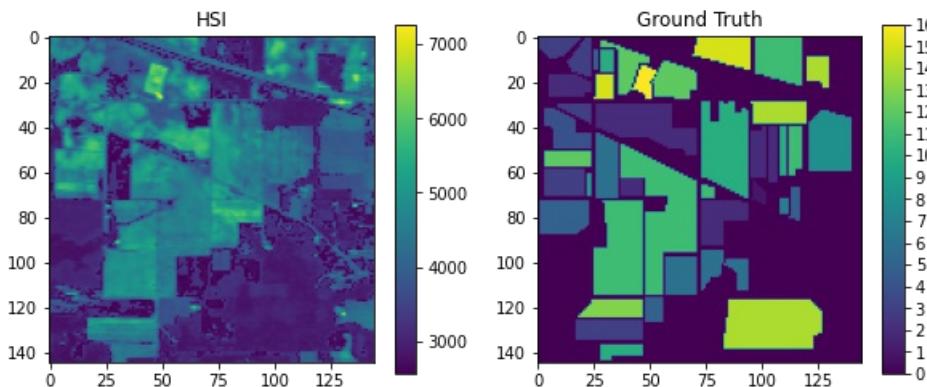
In [3]:

```
img = scipy.io.loadmat('Indian_pines_corrected.mat')['indian_pines_corrected']
img_gt = scipy.io.loadmat('Indian_pines_gt.mat')['indian_pines_gt']
```

In [5]:

```
figr, axis = plt.subplots(1, 2, figsize=(10, 10))
im0 = axis[0].imshow(img[:, :, 20])#, cmap='jet')
axis[0].set_title('HSI')
plt.colorbar(im0, ax=axis[0], shrink=0.4, aspect=16) #, ticks=range(0, 17, 1))

im1 = axis[1].imshow(img_gt) #, cmap='jet')
axis[1].set_title('Ground Truth')
plt.colorbar(im1, ax=axis[1], shrink=0.4, aspect=16, ticks=range(0, 17, 1))
plt.show()
```



In []:

```
img.shape, img_gt.shape
```

Out []:

```
((145, 145, 200), (145, 145))
```

Data Augmentation

Generating Multiple images from available image :

- Rotating image by 90, 180 and 270 deg
- Flipping original and rotated images

In []:

```
img_rot1 = np.rot90(img, 1)
img_gt_rot1 = np.rot90(img_gt, 1)
```

In []:

```
img_rot2 = np.rot90(img, 2)
img_gt_rot2 = np.rot90(img_gt, 2)
```

In []:

```
img_rot3 = np.rot90(img,3)
img_gt_rot3 = np.rot90(img_gt,3)
```

In []:

```
img_rot4 = rotate(img,-45,reshape=False,mode ='reflect',order=0)
img_gt_rot4 = rotate(img_gt,-45,reshape=False,mode ='reflect',order=0)
img_gt_rot4.shape
```

Out[]:

```
(145, 145)
```

In []:

```
img.max(),img_rot1.max(),img_rot2.max(),img_rot3.max(),img_rot4.max()
```

Out[]:

```
(9604, 9604, 9604, 9604, 9604)
```

In []:

```
img_gt.max(),img_gt_rot1.max(),img_gt_rot2.max(),img_gt_rot3.max(),img_gt_rot4.max()
```

Out[]:

```
(16, 16, 16, 16, 16)
```

In []:

```
img.min(),img_rot1.min(),img_rot2.min(),img_rot3.min(),img_rot4.min()
```

Out[]:

```
(955, 955, 955, 955, 955)
```

In []:

```
img_gt.min(),img_gt_rot1.min(),img_gt_rot2.min(),img_gt_rot3.min(),img_gt_rot4.min()
```

Out[]:

```
(0, 0, 0, 0, 0)
```

In []:

```
img_flip = np.fliplr(img)
img_gt_flip = np.fliplr(img_gt)

img_rot1_fp = np.fliplr(img_rot1)
img_gt_rot1_fp = np.fliplr(img_gt_rot1)

img_rot2_fp = np.fliplr(img_rot2)
img_gt_rot2_fp = np.fliplr(img_gt_rot2)

img_rot3_fp = np.fliplr(img_rot3)
img_gt_rot3_fp = np.fliplr(img_gt_rot3)

img_rot4_fp = np.fliplr(img_rot4)
img_gt_rot4_fp = np.fliplr(img_gt_rot4)
```

Generating patches of size 64 x 64 from the augmented images

=> 10 x 10 patches will be generated from one image = 64 cropped images

In []:

```
# image patches of the Augmented Hyperspectral images
img_patches      = np.squeeze(patch.patchify(img,          (64, 64,200), step=9), axis=2)
img_r1_patches   = np.squeeze(patch.patchify(img_rot1,    (64, 64,200), step=9), axis=2)
img_r2_patches   = np.squeeze(patch.patchify(img_rot2,    (64, 64,200), step=9), axis=2)
img_r3_patches   = np.squeeze(patch.patchify(img_rot3,    (64, 64,200), step=9), axis=2)
img_r4_patches   = np.squeeze(patch.patchify(img_rot4,    (64, 64,200), step=9), axis=2)

img_fp_patches   = np.squeeze(patch.patchify(img_flip,     (64, 64,200), step=9), axis=2)
img_r1_fp_patches= np.squeeze(patch.patchify(img_rot1_fp, (64, 64,200), step=9), axis=2)
img_r2_fp_patches= np.squeeze(patch.patchify(img_rot2_fp, (64, 64,200), step=9), axis=2)
img_r3_fp_patches= np.squeeze(patch.patchify(img_rot3_fp, (64, 64,200), step=9), axis=2)
img_r4_fp_patches= np.squeeze(patch.patchify(img_rot4_fp, (64, 64,200), step=9), axis=2)
```

In []:

```
# image patches of the Augmented Ground Truths of Hyperspectral images
img_gt_patches   = patch.patchify(img_gt,      (64, 64), step=9)
img_gt_r1_patches= patch.patchify(img_gt_rot1, (64, 64), step=9)
img_gt_r2_patches= patch.patchify(img_gt_rot2, (64, 64), step=9)
img_gt_r3_patches= patch.patchify(img_gt_rot3, (64, 64), step=9)
img_gt_r4_patches= patch.patchify(img_gt_rot4, (64, 64), step=9)

img_gt_fp_patches= patch.patchify(img_gt_flip,   (64, 64), step=9)
img_gt_r1_fp_patches= patch.patchify(img_gt_rot1_fp, (64, 64), step=9)
img_gt_r2_fp_patches= patch.patchify(img_gt_rot2_fp, (64, 64), step=9)
img_gt_r3_fp_patches= patch.patchify(img_gt_rot3_fp, (64, 64), step=9)
img_gt_r4_fp_patches= patch.patchify(img_gt_rot4_fp, (64, 64), step=9)
```

In []:

```
img_r4_patches.shape, img_gt_r4_patches.shape
```

Out[]:

```
((10, 10, 64, 64, 200), (10, 10, 64, 64))
```

In []:

```
img_r1_fp_patches.shape
```

Out[]:

```
(10, 10, 64, 64, 200)
```

In []:

```
img_patches[5][5][:,:,20].shape
```

Out[]:

```
(64, 64)
```

In []:

```
# img_patches = np.squeeze(img_patches, axis=2).shape
```

In []:

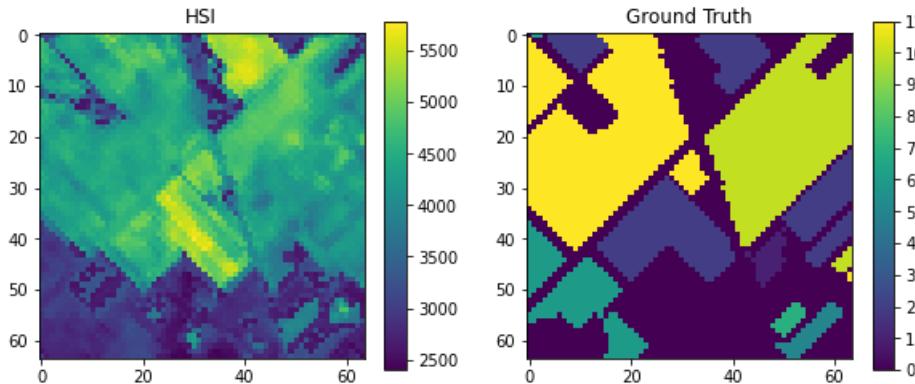
```
# Verifying the augmented data
figr, axis = plt.subplots(1,2, figsize=(10,10))
im0 = axis[0].imshow(img_r4_patches[5][5][:,:,30])#,cmap='jet')
axis[0].set_title('HSI')
```

```

plt.colorbar(im0,ax=axis[0],shrink=0.4,aspect=16) #, ticks=range(0,17,1))

im1 = axis[1].imshow(img_gt_r4_patches[5][5])#,cmap='jet')
axis[1].set_title('Ground Truth')
plt.colorbar(im1,ax=axis[1],shrink=0.4,aspect=16, ticks=range(0,17,1))
# plt.savefig('NeuNet_3_e100.png')
plt.show()

```



Storing images

data are stored in *.mat files (for reuse - to avoid running the augmentation everytime data is required)

In []:

```

# HSI - collection of augmented patches
HSI_AUGM_mat1 = dict()
HSI_AUGM_mat1['img_orig'] = img_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_1.mat',HSI_AUGM_mat1)

HSI_AUGM_mat2 = dict()
HSI_AUGM_mat2['img_rot1'] = img_r1_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_2.mat',HSI_AUGM_mat2)

HSI_AUGM_mat3 = dict()
HSI_AUGM_mat3['img_rot2'] = img_r2_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_3.mat',HSI_AUGM_mat3)

HSI_AUGM_mat4 = dict()
HSI_AUGM_mat4['img_rot3'] = img_r3_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_4.mat',HSI_AUGM_mat4)

HSI_AUGM_mat5 = dict()
HSI_AUGM_mat5['img_rot4'] = img_r4_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_5.mat',HSI_AUGM_mat5)

HSI_AUGM_mat6 = dict()
HSI_AUGM_mat6['img_flp0'] = img_fp_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_6.mat',HSI_AUGM_mat6)

HSI_AUGM_mat7 = dict()
HSI_AUGM_mat7['img_flp1'] = img_r1_fp_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_7.mat',HSI_AUGM_mat7)

HSI_AUGM_mat8 = dict()
HSI_AUGM_mat8['img_flp2'] = img_r2_fp_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_8.mat',HSI_AUGM_mat8)

HSI_AUGM_mat9 = dict()
HSI_AUGM_mat9['img_flp3'] = img_r3_fp_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_9.mat',HSI_AUGM_mat9)

HSI_AUGM_mat10 = dict()
HSI_AUGM_mat10['img_flp4'] = img_r4_fp_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_10.mat',HSI_AUGM_mat10)

```

In []:

```

# Ground Truth patches
HSI_AUGM_GT_mat = dict()
HSI_AUGM_GT_mat['gt_orig'] = img_gt_patches
HSI_AUGM_GT_mat['gt_rot1'] = img_gt_r1_patches
HSI_AUGM_GT_mat['gt_rot2'] = img_gt_r2_patches
HSI_AUGM_GT_mat['gt_rot3'] = img_gt_r3_patches
HSI_AUGM_GT_mat['gt_rot4'] = img_gt_r4_patches
HSI_AUGM_GT_mat['gt_flp0'] = img_gt_fp_patches
HSI_AUGM_GT_mat['gt_flp1'] = img_gt_r1_fp_patches
HSI_AUGM_GT_mat['gt_flp2'] = img_gt_r2_fp_patches
HSI_AUGM_GT_mat['gt_flp3'] = img_gt_r3_fp_patches
HSI_AUGM_GT_mat['gt_flp4'] = img_gt_r4_fp_patches
scipy.io.savemat('Indian_pines_HSI_AUGM_GT.mat', HSI_AUGM_GT_mat)

```

Data Loader for model

Loading the data from *.mat files

The *.mat file data are read and stored in variable.

In [4]:

```

!wget --header="Host: doc-0c-14-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Cookie: A UTH_3hse9g93rcadk3jhj6f018c5j3o1r79s_nonce=r9gnfneb0ci7k" --header="Connection: keep-alive" "https://doc-0c-14-docs.googleusercontent.com/docs/securesc/brr5e7j2b01vas75j784c78kqje1o8uv/nerc8oanodeggildu2ovcgsng2d898u/1653617550000/00176583124175523585/14063734026192868486/1QFviRpVmtM8Q88AuVAn4SGhQ0681tmo3?e=download&ax=ACxEAsay7kPeM8AjC0t09kFA3xWFVFB7zPQyOpJ_r1y9ra_S1eSjKyniU44Kz6fSyUYkb2kuTOkHQxPR9SZ9hm5fBFRRItgNcoOzyiHH8nazPzRXLeHks4rUQ-gbSrV_RnXknBKvvBPZfRAWESJaxjoBwGLFVKOU6hMH8pHYcW_-DgQ-r0iNWxK52VZuHRauhEi4hgB8mlqaTnygm8wx0ko1NQqip35i15MaJNMPx6vwIwOAU8WBKEZux4AUNa4778StQn1tNC40r5-I3pqJH4E-OVEBr7wVyy63HJY_f1x6Kj21PMDm1tYPTnIw-f241YMKlmaNzWBTi7D2HI6r9ZquauKPZndczupSufpHMLyQEcpd6MrE59KId31_f-UPc914OHsu311TgA3S0aEiMhpvrXMztB3ASiu7VFaB-6YoDPvEN07ZAWDvQEO6tOSk2G-K8jUAxhv28Gehr5N7tpVFrzyeCbESatI1YhLDK_ar4gGHXGKmagawlWJ93D511Ndzbr6037gS31g0DQR1m7jljCYTQZHK-jioGzpUBkMdUmn4901ZKq6jRILiEdA8A9XI0woT3uyE5oz7snOQV-SbQF1QDkJ2sYHcp9QiBkg8x1wlmhLj_FlxU9TFC2LySC8cQUOyP-2WA4g_i_YAZ8MyaM-nL49ZDSKb2yX_nI4hOh-NvPOe_p-zmIi-TUzkVcBw190-Fbh4jyc&authuser=0&nonce=r9gnfneb0ci7k&user=14063734026192868486&hash=gsj501gc5vkmvb6i5o77rnuqaad3vd2t" -c -O 'Indian_pines_HSI_AUGM_1to10_gt.zip'
!unzip Indian_pines_HSI_AUGM_1to10_gt.zip

```

```

--2022-05-27 02:13:26-- https://doc-0c-14-docs.googleusercontent.com/docs/securesc/brr5e7j2b01vas75j784c78kqje1o8uv/nerc8oanodeggildu2ovcgsng2d898u/1653617550000/00176583124175523585/14063734026192868486/1QFviRpVmtM8Q88AuVAn4SGhQ0681tmo3?e=download&ax=ACxEAsay7kPeM8AjC0t09kFA3xWFVFB7zPQyOpJ_r1y9ra_S1eSjKyniU44Kz6fSyUYkb2kuTOkHQxPR9SZ9hm5fBFRRItgNcoOzyiHH8nazPzRXLeHks4rUQ-gbSrV_RnXknBKvvBPZfRAWESJaxjoBwGLFVKOU6hMH8pHYcW_-DgQ-r0iNWxK52VZuHRauhEi4hgB8mlqaTnygm8wx0ko1NQqip35i15MaJNMPx6vwIwOAU8WBKEZux4AUNa4778StQn1tNC40r5-I3pqJH4E-OVEBr7wVyy63HJY_f1x6Kj21PMDm1tYPTnIw-f241YMKlmaNzWBTi7D2HI6r9ZquauKPZndczupSufpHMLyQEcpd6MrE59KId31_f-UPc914OHsu311TgA3S0aEiMhpvrXMztB3ASiu7VFaB-6YoDPvEN07ZAWDvQEO6tOSk2G-K8jUAxhv28Gehr5N7tpVFrzyeCbESatI1YhLDK_ar4gGHXGKmagawlWJ93D511Ndzbr6037gS31g0DQR1m7jljCYTQZHK-jioGzpUBkMdUmn4901ZKq6jRIliEdA8A9XI0woT3uyE5oz7snOQV-SbQF1QDkJ2sYHcp9QiBkg8x1wlmhLj_FlxU9TFC2LySC8cQUOyP-2WA4g_i_YAZ8MyaM-nL49ZDSKb2yX_nI4hOh-NvPOe_p-zmIi-TUzkVcBw190-Fbh4jyc&authuser=0&nonce=r9gnfneb0ci7k&user=14063734026192868486&hash=gsj501gc5vkmvb6i5o77rnuqaad3vd2t
Resolving doc-0c-14-docs.googleusercontent.com (doc-0c-14-docs.googleusercontent.com) ... 108.177.98.132, 2607:f8b0:400e:c06::84
Connecting to doc-0c-14-docs.googleusercontent.com (doc-0c-14-docs.googleusercontent.com) |108.177.98.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 412701014 (394M) [application/x-zip-compressed]
Saving to: 'Indian_pines_HSI_AUGM_1to10_gt.zip'

Indian_pines_HSI_AU 100%[=====] 393.58M 100MB/s in 4.0s

```

2022-05-27 02:13:30 (97.9 MB/s) - 'Indian_pines_HSI_AUGM_1to10_gt.zip' saved [412701014/412701014]

```

Archive: Indian_pines_HSI_AUGM_1to10_gt.zip
inflating: Indian_pines_HSI_AUGM_1.mat
inflating: Indian_pines_HSI_AUGM_10.mat
inflating: Indian_pines_HSI_AUGM_2.mat
inflating: Indian_pines_HSI_AUGM_3.mat
inflating: Indian_pines_HSI_AUGM_4.mat
inflating: Indian_pines_HSI_AUGM_5.mat
inflating: Indian_pines_HSI_AUGM_6.mat
inflating: Indian_pines_HSI_AUGM_7.mat

```

```
inflating: Indian_pines_HSI_AUGM_8.mat
inflating: Indian_pines_HSI_AUGM_9.mat
inflating: Indian_pines_HSI_AUGM_GT.mat
```

In [4]:

```
HSI_AUGM_1 = scipy.io.loadmat('Indian_pines_HSI_AUGM_1.mat')['img_orig']
HSI_AUGM_2 = scipy.io.loadmat('Indian_pines_HSI_AUGM_2.mat')['img_rot1']
HSI_AUGM_3 = scipy.io.loadmat('Indian_pines_HSI_AUGM_3.mat')['img_rot2']
HSI_AUGM_4 = scipy.io.loadmat('Indian_pines_HSI_AUGM_4.mat')['img_rot3']
HSI_AUGM_5 = scipy.io.loadmat('Indian_pines_HSI_AUGM_5.mat')['img_rot4']
HSI_AUGM_6 = scipy.io.loadmat('Indian_pines_HSI_AUGM_6.mat')['img_flp0']
HSI_AUGM_7 = scipy.io.loadmat('Indian_pines_HSI_AUGM_7.mat')['img_flp1']
HSI_AUGM_8 = scipy.io.loadmat('Indian_pines_HSI_AUGM_8.mat')['img_flp2']
HSI_AUGM_9 = scipy.io.loadmat('Indian_pines_HSI_AUGM_9.mat')['img_flp3']
HSI_AUGM_10 = scipy.io.loadmat('Indian_pines_HSI_AUGM_10.mat')['img_flp4']
```

In [5]:

```
# list to generate the dataset
img_patch_list = [HSI_AUGM_1,
                  HSI_AUGM_2,
                  HSI_AUGM_3,
                  HSI_AUGM_4,
                  HSI_AUGM_5,
                  HSI_AUGM_6,
                  HSI_AUGM_7,
                  HSI_AUGM_8,
                  HSI_AUGM_9,
                  HSI_AUGM_10]
```

In [6]:

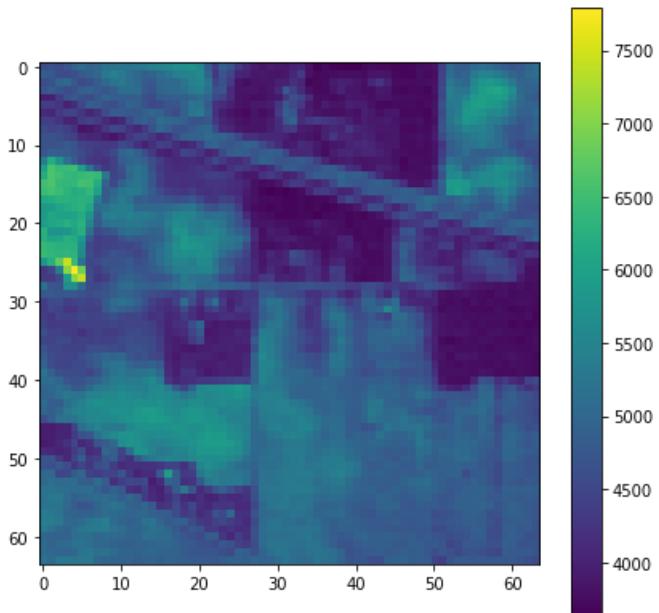
```
HSI_AUGM_1.shape
```

Out[6]:

```
(10, 10, 64, 64, 200)
```

In [7]:

```
# Example plot
plt.figure(figsize=(7,7))
plt.imshow(HSI_AUGM_1[0][5][:,:,10])
plt.colorbar()
plt.show()
```



In [8]:

```
HSI_GT_AUGM_mat = scipy.io.loadmat('Indian_pines_HSI_AUGM_GT.mat')
```

In [9]:

```
list(HSI_GT_AUGM_mat.keys())[3:]
```

Out[9]:

```
['gt_orig',
 'gt_rot1',
 'gt_rot2',
 'gt_rot3',
 'gt_rot4',
 'gt_flp0',
 'gt_flp1',
 'gt_flp2',
 'gt_flp3',
 'gt_flp4']
```

In [10]:

```
img_gt_patch_list = []
for key in list(HSI_GT_AUGM_mat.keys())[3:]:
    img_gt_patch_list.append(HSI_GT_AUGM_mat[key])
```

In [11]:

```
img_gt_patch_list[1].shape
```

Out[11]:

```
(10, 10, 64, 64)
```

In [12]:

```
img.reshape(-1, img.shape[-1]).shape
```

Out[12]:

```
(21025, 200)
```

Removing the bands which have high correlation(0.99) with other features

In [13]:

```
# Reference for correlation feature filtering : https://sachinbu.medium.com/hyperspectral-image-segmentation-21432965e138
corr_feat_list = [7, 8, 9, 15, 24, 27, 28, 38, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 58,
64, 65, 66, 67, 68, 69, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123,
124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 147, 148, 149,
151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 1
71, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190]
```

In [14]:

```
img_patch_list_new = []
for patchs in img_patch_list:
    filtered_patchs = np.delete(patchs, corr_feat_list, -1)
    img_patch_list_new.append(filtered_patchs)
```

In [15]:

```
img_patch_list_new[9].shape
```

Out[15]:

```
(10, 10, 64, 64, 95)
```

In [16]:

```
# Deleting variable to make space for other data
del img_patch_list
del HSI_AUGM_1
del HSI_AUGM_2
del HSI_AUGM_3
del HSI_AUGM_4
del HSI_AUGM_5
del HSI_AUGM_6
del HSI_AUGM_7
del HSI_AUGM_8
del HSI_AUGM_9
del HSI_AUGM_10
```

Standardization

Standaradizing the values of the image matrix for each band

In [17]:

```
# Removing 105 features before standardizing data
img_filtered = np.delete(img,corr_feat_list,-1)
```

In [18]:

```
#Standardizing the data
Std_scaler = StandardScaler()
Std_scaler.fit(img_filtered.reshape(-1,img_filtered.shape[-1]))
```

Out[18]:

```
StandardScaler()
```

Creating Dataset to have collection of images instead of patches

In [19]:

```
# Generating Image dataset seperating the single 64x64x95 patch from patch grid (10,10,64,64,95) after
#standardising
image_dataset = []
for patchs in img_patch_list_new:
    for i in range(patchs.shape[0]):
        for j in range(patchs.shape[1]):
            single_patch = patchs[i][j]
            single_patch = Std_scaler.transform(single_patch.reshape(-1,single_patch.shape[-1])).reshape(single_patch.shape)
            image_dataset.append(single_patch)
```

In [20]:

```
image_dataset = np.array(image_dataset)
image_dataset.shape
```

Out[20]:

```
(1000, 64, 64, 95)
```

In [21]:

```
# Generating Groundtruth dataset seperating the single 64x64 patch from patch grid (10,10,64,64)
gt_dataset = []
for patchs in img_gt_patch_list:
    for i in range(patchs.shape[0]):
        for j in range(patchs.shape[1]):
            gt_dataset.append(patchs[i][j])
```

In [22]:

```
gt_dataset = np.array(gt_dataset)
gt_dataset.shape
```

Out[22]:

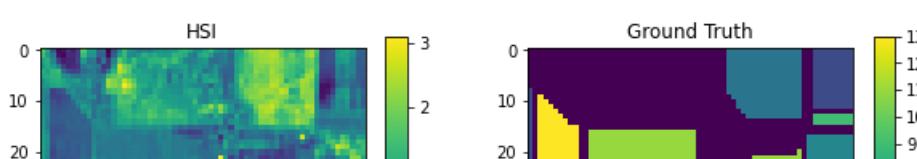
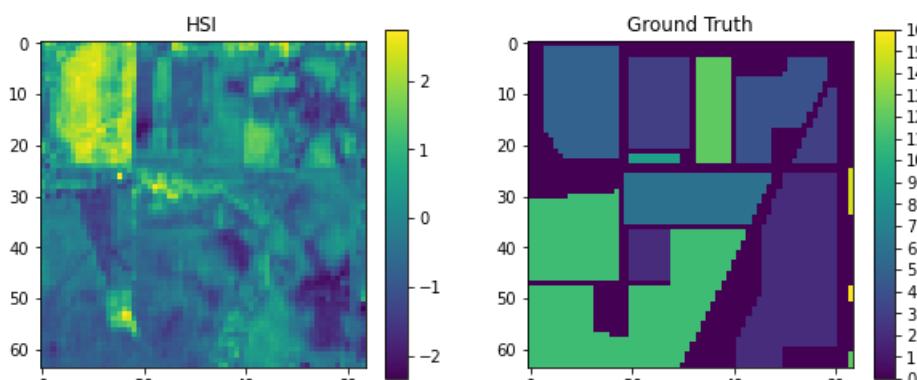
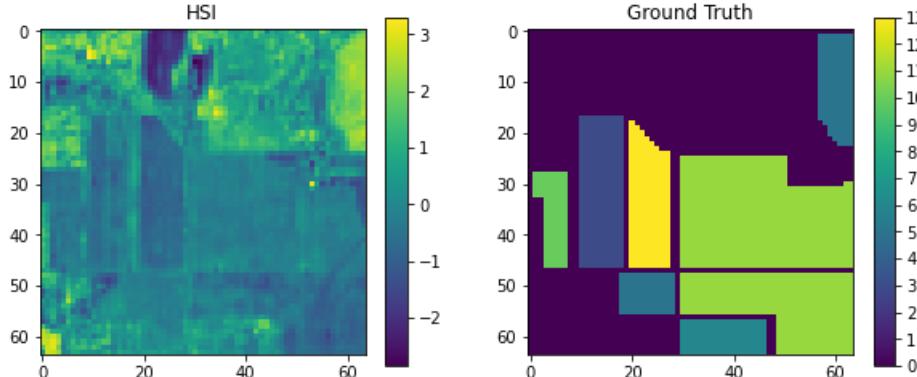
```
(1000, 64, 64)
```

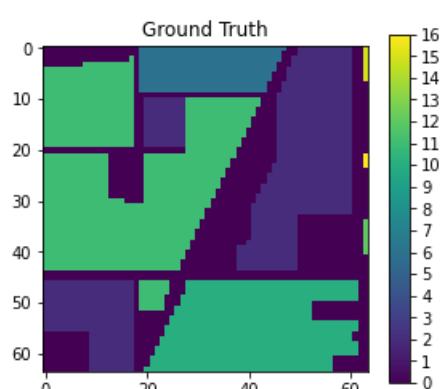
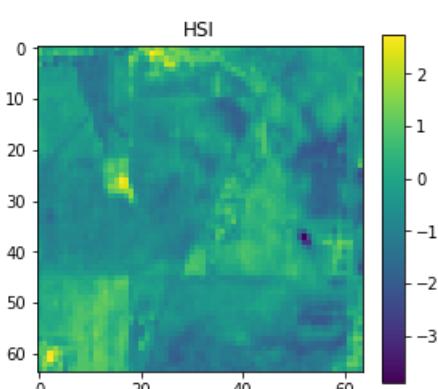
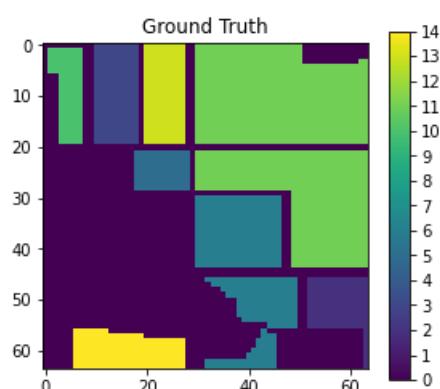
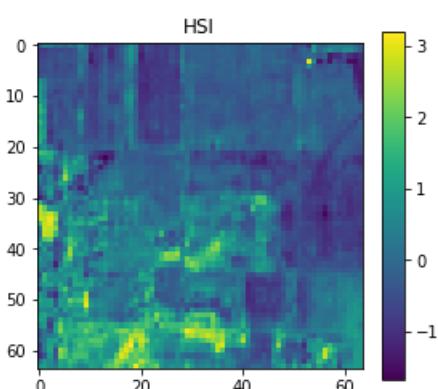
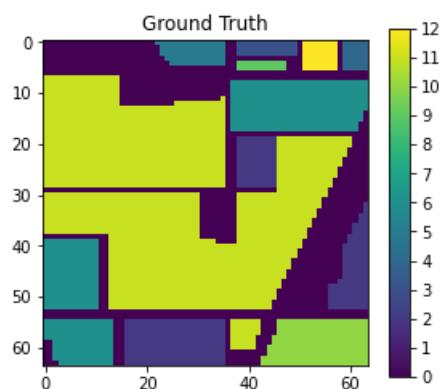
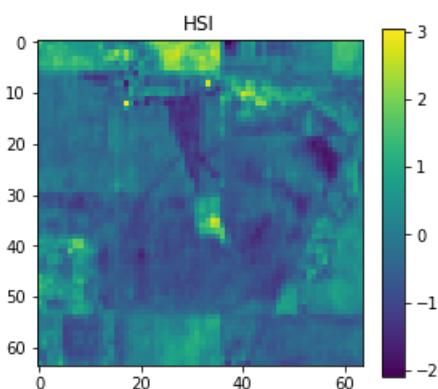
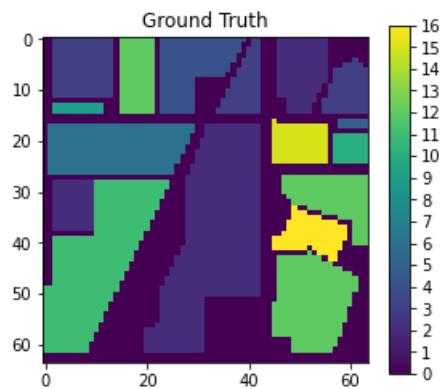
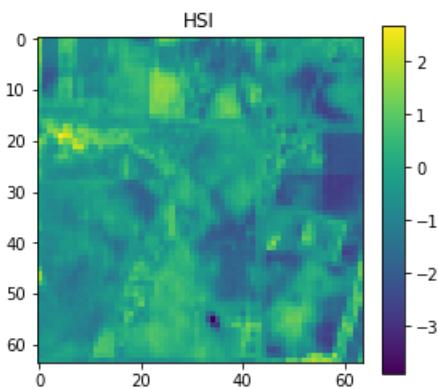
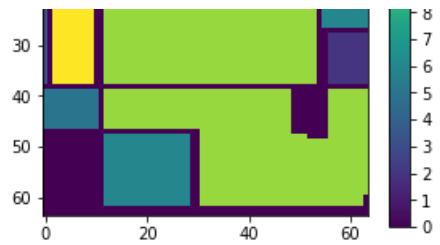
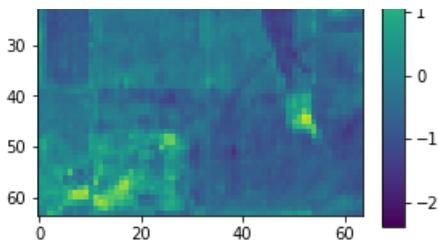
Dataset Review

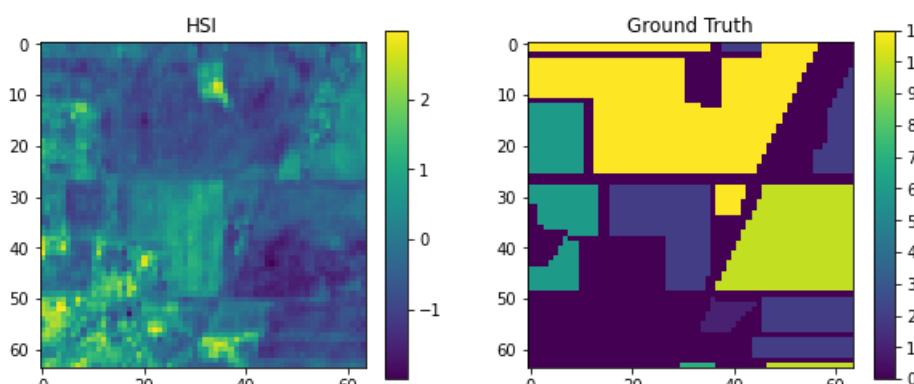
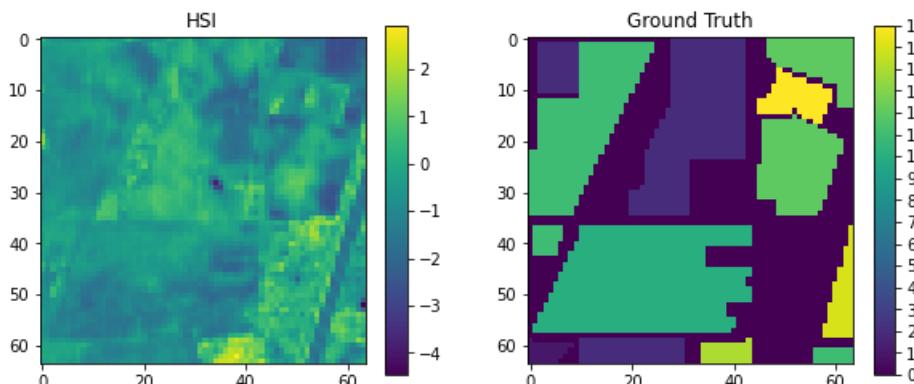
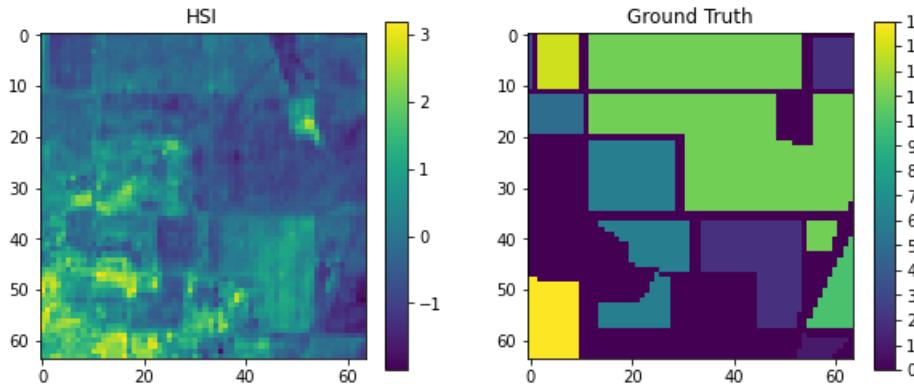
In [23]:

```
for i in range(100,120,2):
    figr, axis = plt.subplots(1,2, figsize=(10,10))
    im0 = axis[0].imshow(image_dataset[i*3][:,:,30])#,cmap='jet')
    axis[0].set_title('HSI')
    plt.colorbar(im0, ax=axis[0], shrink=0.4, aspect=16) #, ticks=range(0,17,1))

    im1 = axis[1].imshow(gt_dataset[i*3])#,cmap='jet')
    axis[1].set_title('Ground Truth')
    plt.colorbar(im1, ax=axis[1], shrink=0.4, aspect=16, ticks=range(0,17,1))
    plt.show()
```







Data loader definition

Dataset loader used to pass data for training the model

In [24]:

```
class Dataset:
    def __init__(self, images, gt_images, classes, test_set):
        ''' Dataset to have list of train/test data. image loaded upon calling __getitem__ function'''
        self.image = images
        self.gt = gt_images
        self.classes = classes # list of class label/values
        self.test_set = test_set # Boolean to differentiate train and test data

    def __getitem__(self, i):
        image = self.image[i]

        gt_image = [(self.gt[i]==c) for c in self.classes]
        gt_image = np.stack(gt_image, axis=-1).astype('float')

        return image, gt_image

    def __len__(self):
        return len(self.image)
```

In [25]:

```
class Dataloder(tf.keras.utils.Sequence):
    def __init__(self, dataset, batch_size=1, shuffle=False):
        ''' This class loads data in batches while training the model'''
        self.dataset = dataset
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.indexes = np.arange(len(dataset))

    def __getitem__(self, i):
        # collect batch data
        start = i * self.batch_size
        stop = (i + 1) * self.batch_size
        data = []
        for j in range(start, stop):
            data.append(self.dataset[j])

        batch = [np.stack(samples, axis=0) for samples in zip(*data)]
        return tuple(batch)

    def __len__(self):
        return len(self.indexes) // self.batch_size

    def on_epoch_end(self):
        if self.shuffle:
            self.indexes = np.random.permutation(self.indexes)
```

Verify the dataset class and dataloader class

In [26]:

```
test = Dataset(image_dataset, gt_dataset, list(range(0,17)),0)
```

In [27]:

```
ex = test.__getitem__(150)
```

In [28]:

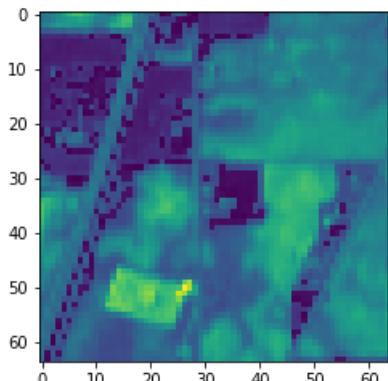
```
ex[1][:,:,10].any()
```

Out[28]:

True

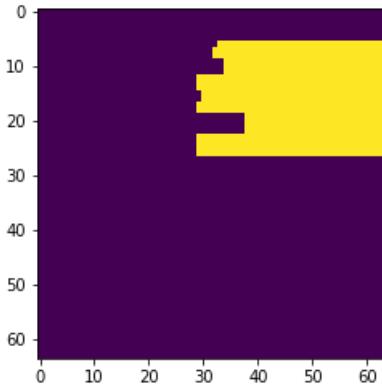
In [29]:

```
plt.imshow(ex[0][:,:,15])
plt.show()
```



In [30]:

```
plt.imshow(ex[1][:,:,10])
plt.show()
```



In [31]:

```
loader = Dataloder(test, batch_size=5, shuffle=False)
```

In [32]:

```
test_batch = loader.__getitem__(50)
```

In [33]:

```
test_batch[0].shape, test_batch[1].shape
```

Out[33]:

```
((5, 64, 64, 95), (5, 64, 64, 17))
```

Train and Test split of data

Data are split into 80% train and 20% test

In [34]:

```
from sklearn.model_selection import train_test_split
X = image_dataset
y = gt_dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=30)
```

In [35]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[35]:

```
((800, 64, 64, 95), (200, 64, 64, 95), (800, 64, 64), (200, 64, 64))
```

Dataset generation

In [36]:

```
# Dataloader for training and testing
CLASSES = list(range(17))
```

```

train_dataset = Dataset(X_train,y_train, classes=CLASSES,test_set = 0)
test_dataset = Dataset(X_test,y_test, classes=CLASSES,test_set = 1)

BATCH_SIZE=10
train_dataloader = Dataloder(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
test_dataloader = Dataloder(test_dataset, batch_size=BATCH_SIZE, shuffle=True)

print('train_dataloader image size :',train_dataloader[0][0].shape)
print('train_dataloader ground truth size :',train_dataloader[0][1].shape)
assert train_dataloader[0][0].shape == (BATCH_SIZE, 64, 64, 95)
assert train_dataloader[0][1].shape == (BATCH_SIZE, 64, 64, 17)

```

```

train_dataloader image size : (10, 64, 64, 95)
train_dataloader ground truth size : (10, 64, 64, 17)

```

Confusion matrix for prediction evaluation

In [37]:

```

from sklearn.metrics import confusion_matrix, f1_score, cohen_kappa_score
import seaborn as sb

```

In [38]:

```

# code reference: appliedaicourse.com case studies
def plot_confusion_matrix_2(test_y, predict_y):
    """
    This function generates the confusion matrix.
    Also evaluates the micro F1 score and Average Accuracy for the predictions.
    """

    print('Confusion / Precision / Recall matrix')
    C = confusion_matrix(test_y, predict_y)
    # print("Number of misclassified points ",(len(test_y)-np.trace(C))/len(test_y)*100)
    print("Percentage of misclassified points ",(np.sum(C)-np.trace(C))/np.sum(C)*100)
    # C = 17x17 matrix, each cell (i,j) represents number of points of class i are predicted class j

    #Precision matrix
    A =(C/C.sum(axis=0))
    #divid each element of the confusion matrix with the sum of elements in that column

    #Recall matrix
    B =(((C.T)/(C.sum(axis=1))).T)
    #divid each element of the confusion matrix with the sum of elements in that row

    labels = list(range(0,17,1))
    cmap=sb.light_palette("green")
    # representing C in heatmap format
    print("-"*50, "Confusion matrix", "-"*50)
    plt.figure(figsize=(16,8))
    sb.heatmap(C, annot=True, cmap=cmap, fmt=".1f", xticklabels=labels[0:17], yticklabels=labels[0:17])
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    # representing A in heatmap format
    print("-"*50, "Precision matrix", "-"*50)
    plt.figure(figsize=(16,8))
    sb.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels[0:17], yticklabels=labels[0:17])
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("Sum of columns in precision matrix",A.sum(axis=0))

    # representing B in heatmap format
    print("-"*50, "Recall matrix", "-"*50)
    plt.figure(figsize=(16,8))
    sb.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels[0:17], yticklabels=labels[0:17])
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("Sum of rows in recall matrix",B.sum(axis=1))

```

```

#sum of all True positives
TP = np.trace(C)

#sum of all True positives and False Positives
TP_FP = np.sum(C.sum(axis=1))

#sum of all True positives and False Negatives
TP_NP = np.sum(C.sum(axis=0))

#micro F1 score evaluation
micro_Pr = TP / TP_FP
micro_Re = TP / TP_NP
micro_F1 = 2 * (micro_Pr * micro_Re) / (micro_Pr + micro_Re)

print('\n micro F1 score : ', micro_F1)

AA = np.trace(B) / 17
print('\n Average Accuracy : ', AA)

```

A2FPN Models

Model 1 - Full model trained from scratch

Here model is defined using segmentation_models module as base model.

Output of FPN model at concatenation stage is considered as input to Attention Module as described in the paper : 'A2-FPN for Semantic Segmentation of Fine-Resolution Remotely Sensed Images' (arXiv:2102.07997v3)

Model Definition

In [39]:

```
!pip install -U segmentation-models
```

```

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: segmentation-models in /usr/local/lib/python3.7/dist-packages (1.0.1)
Requirement already satisfied: keras-applications<=1.0.8,>=1.0.7 in /usr/local/lib/python3.7/dist-packages (from segmentation-models) (1.0.8)
Requirement already satisfied: efficientnet==1.0.0 in /usr/local/lib/python3.7/dist-packages (from segmentation-models) (1.0.0)
Requirement already satisfied: image-classifiers==1.0.0 in /usr/local/lib/python3.7/dist-packages (from segmentation-models) (1.0.0)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages (from efficientnet==1.0.0->segmentation-models) (0.18.3)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from keras-applications<=1.0.8,>=1.0.7->segmentation-models) (3.1.0)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages (from keras-applications<=1.0.8,>=1.0.7->segmentation-models) (1.21.6)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py->keras-applications<=1.0.8,>=1.0.7->segmentation-models) (1.5.2)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models) (2021.11.2)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models) (2.4.1)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models) (2.6.3)
Requirement already satisfied: scipy>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models) (1.4.1)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models) (1.3.0)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models) (3.2.2)
Requirement already satisfied: pillow!=7.1.0,!>=7.1.1,>=4.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models) (7.1.2)
Requirement already satisfied: pyparsing!=2.0.4,!>=2.1.2,!>=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (0.11.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from mat

```

```

Requirement already satisfied: pillow >=6.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (1.4.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (4.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (1.15.0)

```

In [40]:

```

# we are importing the pretrained unet from the segmentation models
# https://github.com/qubvel/segmentation_models
import tensorflow
import tensorflow as tf
import segmentation_models as sm
sm.set_framework('tf.keras')
from segmentation_models import FPN
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Conv2DTranspose, concatenate, Cropping2D, ZeroPadding2D
from tensorflow.keras.layers import UpSampling2D, Reshape, Dropout
from tensorflow.keras.models import Model
from segmentation_models.metrics import iou_score
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceLROnPlateau
tensorflow.keras.backend.set_image_data_format('channels_last')

```

Segmentation Models: using `keras` framework.

FPN model without attention for reference

In [41]:

```
# del FPN_base_model
```

In []:

```

FPN_base_model = FPN(backbone_name='resnet34',
                      input_shape=(64, 64, 95),
                      classes=17,
                      activation='softmax',
                      weights=None, encoder_weights=None, #'imagenet',
                      encoder_freeze=False,
                      encoder_features='default',
                      pyramid_block_filters=64,
                      pyramid_use_batchnorm=True,
                      pyramid_aggregation='concat', pyramid_dropout=None)

```

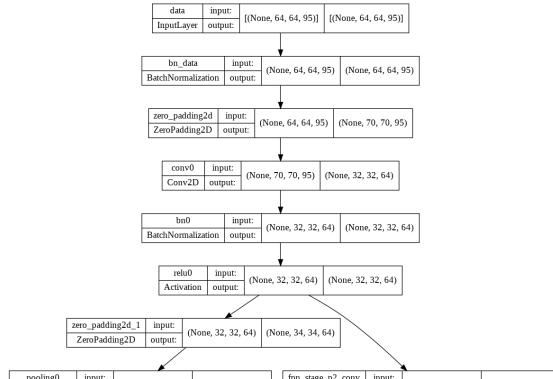
In []:

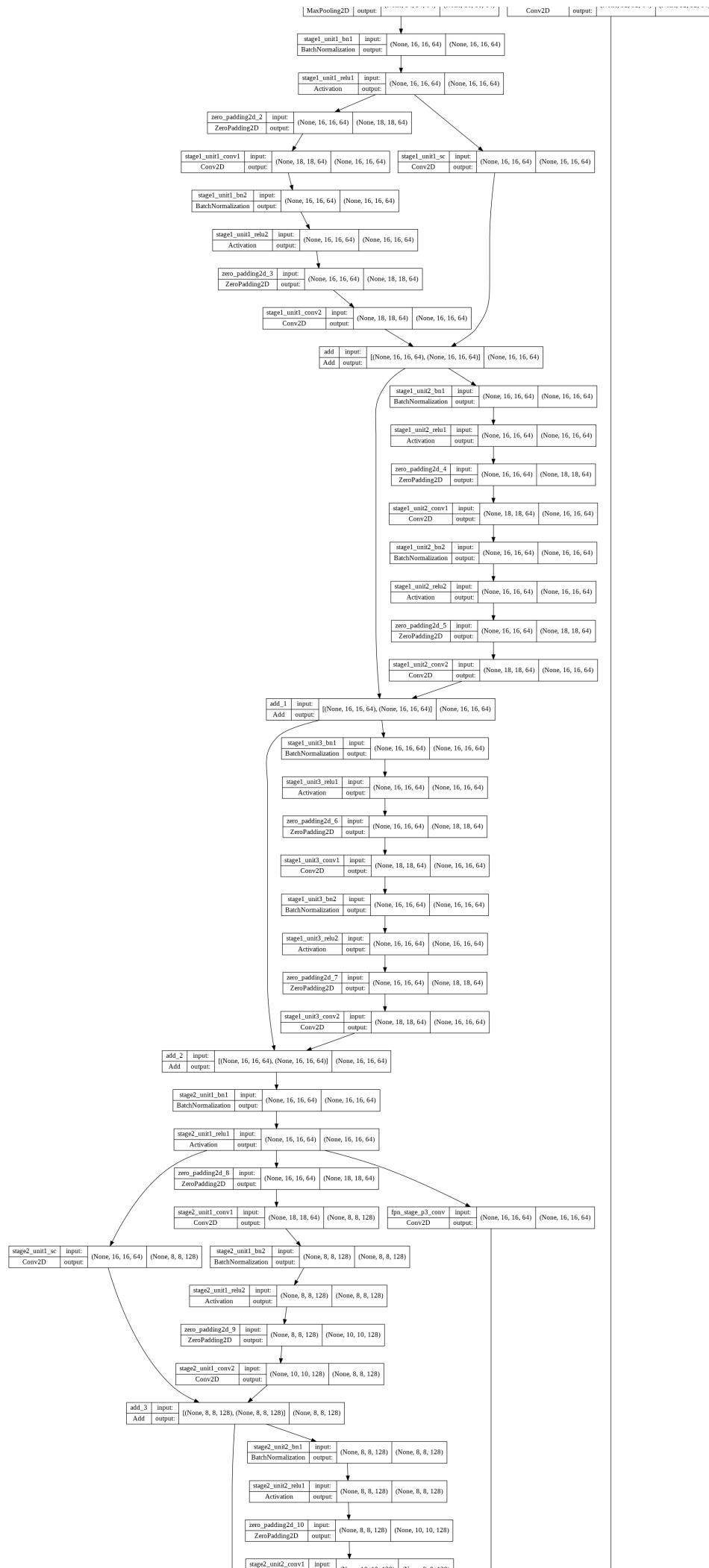
```

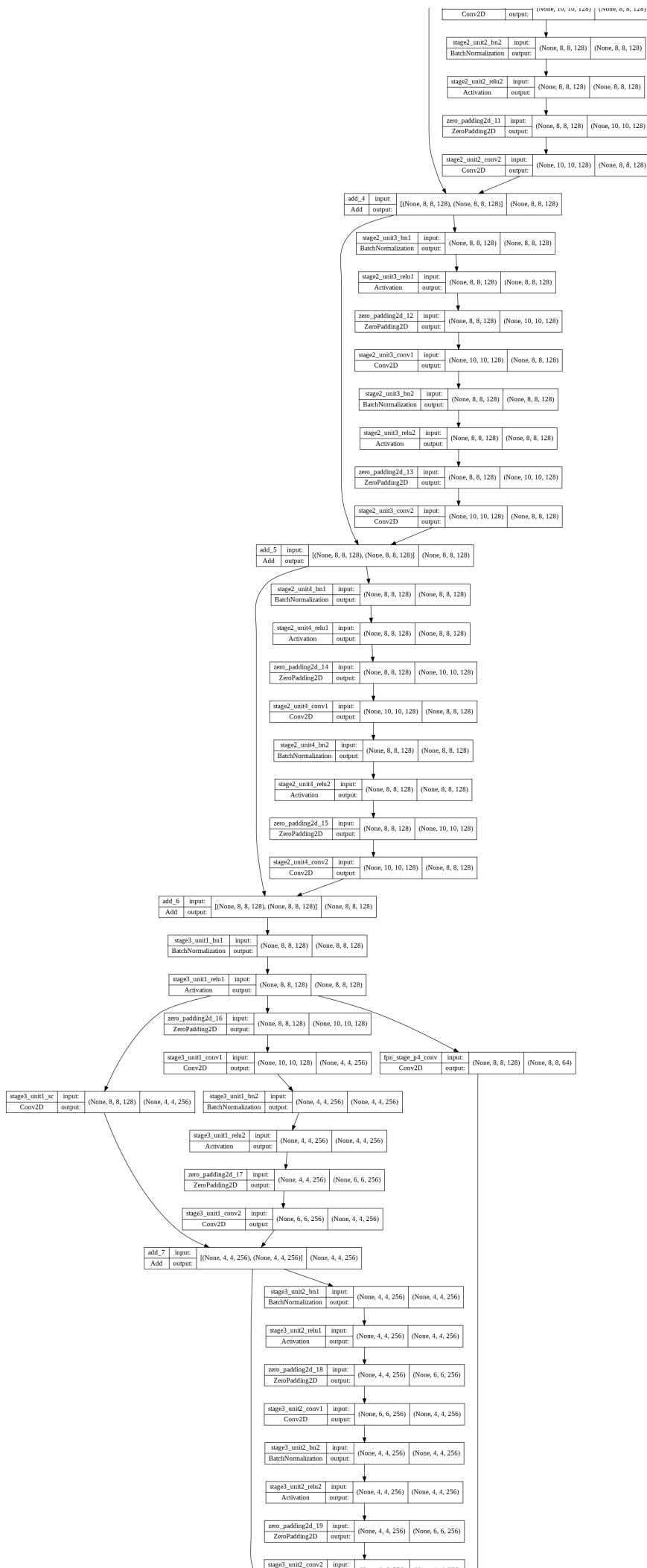
tensorflow.keras.utils.plot_model(FPN_base_model, to_file='model_A2FPN.png', show_shapes=True, show_layer_names=True, expand_nested=False,
                                    rankdir='TB')

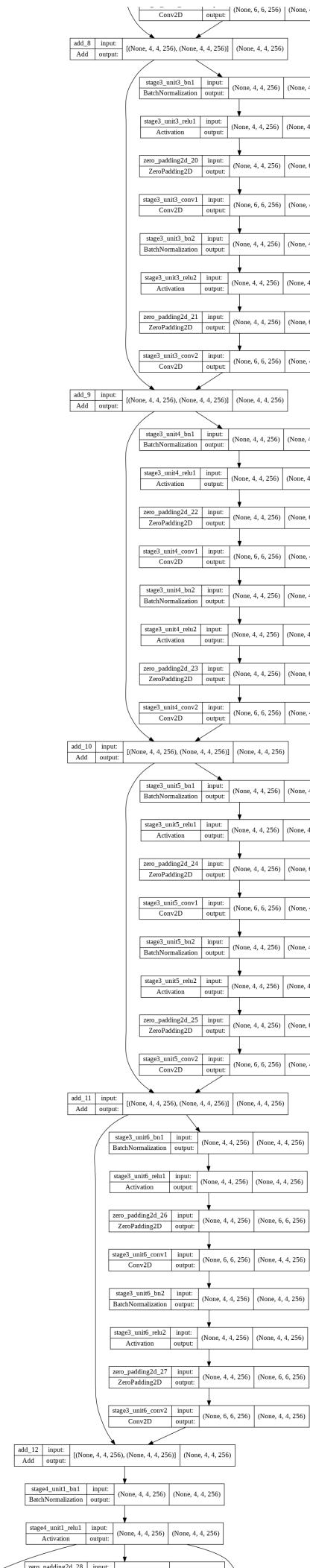
```

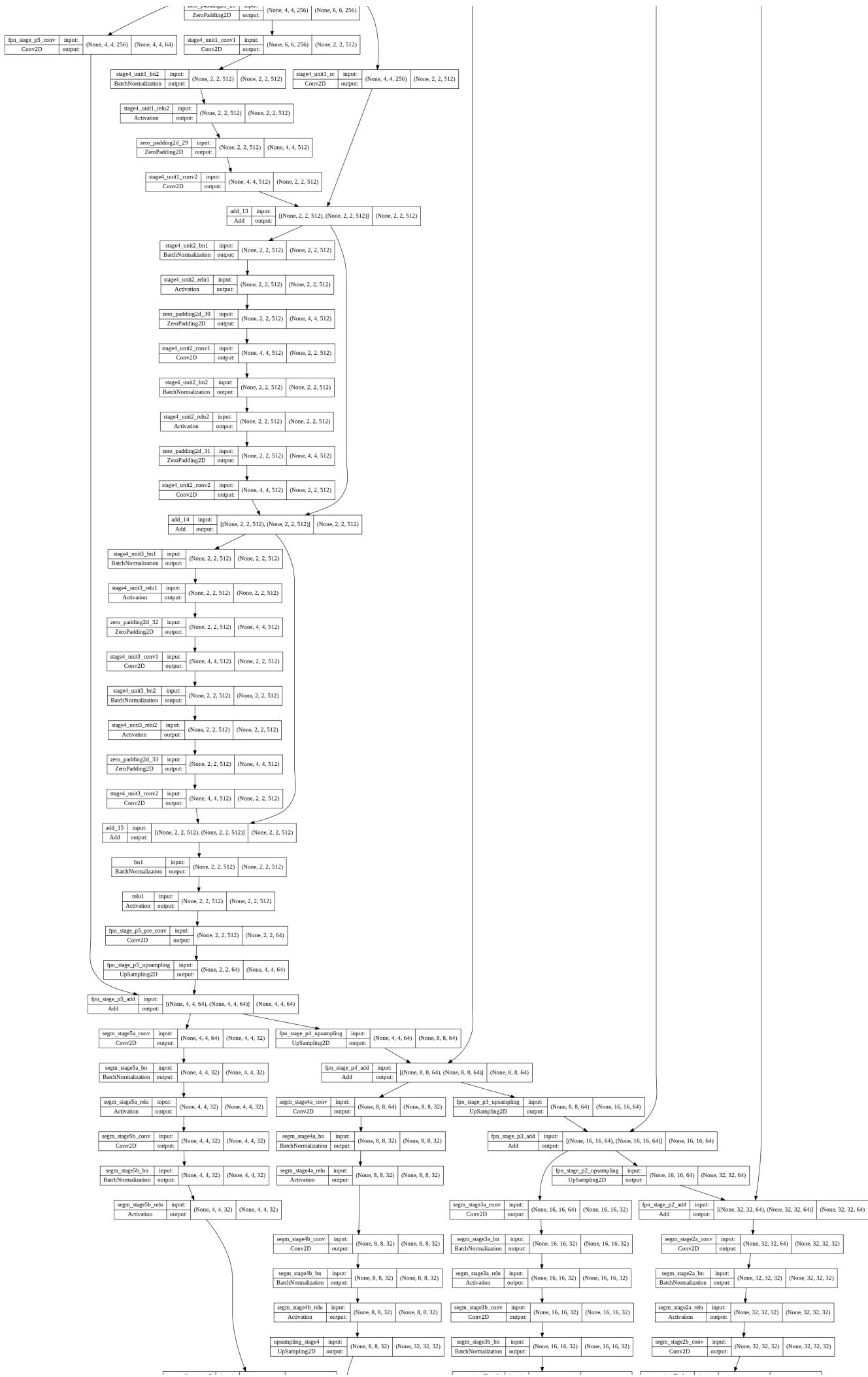
Out[]:

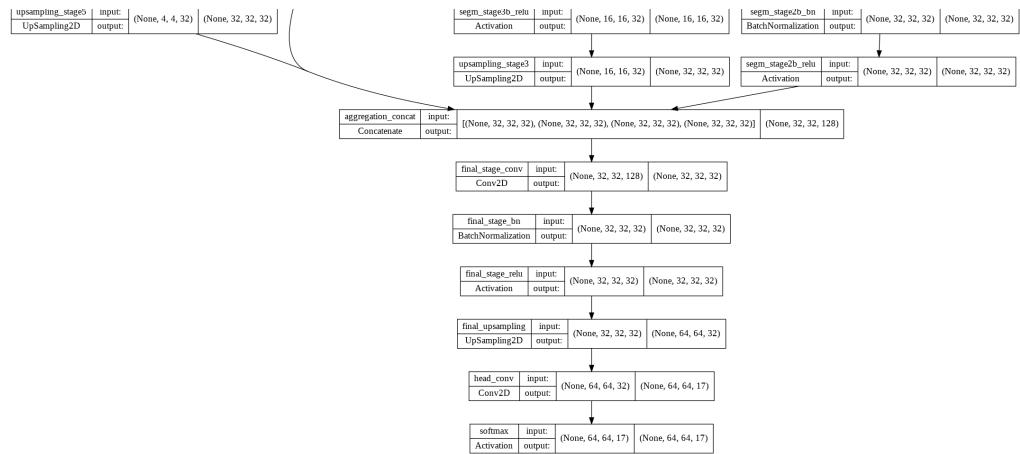












In []:

```
FPN_base_model.summary()
```

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
data (InputLayer)	[(None, 64, 64, 95)]	0	[]
bn_data (BatchNormalization)	(None, 64, 64, 95)	285	['data[0][0]']
zero_padding2d (ZeroPadding2D)	(None, 70, 70, 95)	0	['bn_data[0][0]']
conv0 (Conv2D)	(None, 32, 32, 64)	297920	['zero_padding2d[0][0]']
bn0 (BatchNormalization)	(None, 32, 32, 64)	256	['conv0[0][0]']
relu0 (Activation)	(None, 32, 32, 64)	0	['bn0[0][0]']
zero_padding2d_1 (ZeroPadding2D)	(None, 34, 34, 64)	0	['relu0[0][0]']
pooling0 (MaxPooling2D)	(None, 16, 16, 64)	0	['zero_padding2d_1[0][0]']
stage1_unit1_bn1 (BatchNormalization)	(None, 16, 16, 64)	256	['pooling0[0][0]']
stage1_unit1_relu1 (Activation)	(None, 16, 16, 64)	0	['stage1_unit1_bn1[0][0]']
zero_padding2d_2 (ZeroPadding2D)	(None, 18, 18, 64)	0	['stage1_unit1_relu1[0][0]']
stage1_unit1_conv1 (Conv2D)	(None, 16, 16, 64)	36864	['zero_padding2d_2[0][0]']
stage1_unit1_bn2 (BatchNormalization)	(None, 16, 16, 64)	256	['stage1_unit1_conv1[0][0]']
stage1_unit1_relu2 (Activation)	(None, 16, 16, 64)	0	['stage1_unit1_bn2[0][0]']
zero_padding2d_3 (ZeroPadding2D)	(None, 18, 18, 64)	0	['stage1_unit1_relu2[0][0]']
stage1_unit1_conv2 (Conv2D)	(None, 16, 16, 64)	36864	['zero_padding2d_3[0][0]']
stage1_unit1_sc (Conv2D)	(None, 16, 16, 64)	4096	['stage1_unit1_relu1[0][0]']
add (Add)	(None, 16, 16, 64)	0	['stage1_unit1_conv2[0][0]', 'stage1_unit1_sc[0][0]']
stage1_unit2_bn1 (BatchNormalization)	(None, 16, 16, 64)	256	['add[0][0]']
stage1_unit2_relu1 (Activation)	(None, 16, 16, 64)	0	['stage1_unit2_bn1[0][0]']

)				
zero_padding2d_4 (ZeroPadding2D)	(None, 18, 18, 64)	0		['stage1_unit2_relu1[0][0]']
stage1_unit2_conv1 (Conv2D)	(None, 16, 16, 64)	36864		['zero_padding2d_4[0][0]']
stage1_unit2_bn2 (BatchNormalization)	(None, 16, 16, 64)	256		['stage1_unit2_conv1[0][0]']
stage1_unit2_relu2 (Activation)	(None, 16, 16, 64)	0		['stage1_unit2_bn2[0][0]']
zero_padding2d_5 (ZeroPadding2D)	(None, 18, 18, 64)	0		['stage1_unit2_relu2[0][0]']
stage1_unit2_conv2 (Conv2D)	(None, 16, 16, 64)	36864		['zero_padding2d_5[0][0]']
add_1 (Add)	(None, 16, 16, 64)	0		['stage1_unit2_conv2[0][0]', 'add[0][0]']
stage1_unit3_bn1 (BatchNormalization)	(None, 16, 16, 64)	256		['add_1[0][0]']
stage1_unit3_relu1 (Activation)	(None, 16, 16, 64)	0		['stage1_unit3_bn1[0][0]']
zero_padding2d_6 (ZeroPadding2D)	(None, 18, 18, 64)	0		['stage1_unit3_relu1[0][0]']
stage1_unit3_conv1 (Conv2D)	(None, 16, 16, 64)	36864		['zero_padding2d_6[0][0]']
stage1_unit3_bn2 (BatchNormalization)	(None, 16, 16, 64)	256		['stage1_unit3_conv1[0][0]']
stage1_unit3_relu2 (Activation)	(None, 16, 16, 64)	0		['stage1_unit3_bn2[0][0]']
zero_padding2d_7 (ZeroPadding2D)	(None, 18, 18, 64)	0		['stage1_unit3_relu2[0][0]']
stage1_unit3_conv2 (Conv2D)	(None, 16, 16, 64)	36864		['zero_padding2d_7[0][0]']
add_2 (Add)	(None, 16, 16, 64)	0		['stage1_unit3_conv2[0][0]', 'add_1[0][0]']
stage2_unit1_bn1 (BatchNormalization)	(None, 16, 16, 64)	256		['add_2[0][0]']
stage2_unit1_relu1 (Activation)	(None, 16, 16, 64)	0		['stage2_unit1_bn1[0][0]']
zero_padding2d_8 (ZeroPadding2D)	(None, 18, 18, 64)	0		['stage2_unit1_relu1[0][0]']
stage2_unit1_conv1 (Conv2D)	(None, 8, 8, 128)	73728		['zero_padding2d_8[0][0]']
stage2_unit1_bn2 (BatchNormalization)	(None, 8, 8, 128)	512		['stage2_unit1_conv1[0][0]']
stage2_unit1_relu2 (Activation)	(None, 8, 8, 128)	0		['stage2_unit1_bn2[0][0]']
zero_padding2d_9 (ZeroPadding2D)	(None, 10, 10, 128)	0		['stage2_unit1_relu2[0][0]']
stage2_unit1_conv2 (Conv2D)	(None, 8, 8, 128)	147456		['zero_padding2d_9[0][0]']
stage2_unit1_sc (Conv2D)	(None, 8, 8, 128)	8192		['stage2_unit1_relu1[0][0]']
add_3 (Add)	(None, 8, 8, 128)	0		['stage2_unit1_conv2[0][0]', 'stage2_unit1_sc[0][0]']
stage2_unit2_bn1 (BatchNormalization)	(None, 8, 8, 128)	512		['add_3[0][0]']
stage2_unit2_relu1 (Activation)	(None, 8, 8, 128)	0		['stage2_unit2_bn1[0][0]']

)					
zero_padding2d_10 (ZeroPadding2D)	(None, 10, 10, 128)	0			['stage2_unit2_relu1[0][0]']
stage2_unit2_conv1 (Conv2D)	(None, 8, 8, 128)	147456			['zero_padding2d_10[0][0]']
stage2_unit2_bn2 (BatchNormalization)	(None, 8, 8, 128)	512			['stage2_unit2_conv1[0][0]']
stage2_unit2_relu2 (Activation)	(None, 8, 8, 128)	0			['stage2_unit2_bn2[0][0]']
)					
zero_padding2d_11 (ZeroPadding2D)	(None, 10, 10, 128)	0			['stage2_unit2_relu2[0][0]']
stage2_unit2_conv2 (Conv2D)	(None, 8, 8, 128)	147456			['zero_padding2d_11[0][0]']
add_4 (Add)	(None, 8, 8, 128)	0			['stage2_unit2_conv2[0][0]', 'add_3[0][0]']
stage2_unit3_bn1 (BatchNormalization)	(None, 8, 8, 128)	512			['add_4[0][0]']
stage2_unit3_relu1 (Activation)	(None, 8, 8, 128)	0			['stage2_unit3_bn1[0][0]']
)					
zero_padding2d_12 (ZeroPadding2D)	(None, 10, 10, 128)	0			['stage2_unit3_relu1[0][0]']
stage2_unit3_conv1 (Conv2D)	(None, 8, 8, 128)	147456			['zero_padding2d_12[0][0]']
stage2_unit3_bn2 (BatchNormalization)	(None, 8, 8, 128)	512			['stage2_unit3_conv1[0][0]']
stage2_unit3_relu2 (Activation)	(None, 8, 8, 128)	0			['stage2_unit3_bn2[0][0]']
)					
zero_padding2d_13 (ZeroPadding2D)	(None, 10, 10, 128)	0			['stage2_unit3_relu2[0][0]']
stage2_unit3_conv2 (Conv2D)	(None, 8, 8, 128)	147456			['zero_padding2d_13[0][0]']
add_5 (Add)	(None, 8, 8, 128)	0			['stage2_unit3_conv2[0][0]', 'add_4[0][0]']
stage2_unit4_bn1 (BatchNormalization)	(None, 8, 8, 128)	512			['add_5[0][0]']
stage2_unit4_relu1 (Activation)	(None, 8, 8, 128)	0			['stage2_unit4_bn1[0][0]']
)					
zero_padding2d_14 (ZeroPadding2D)	(None, 10, 10, 128)	0			['stage2_unit4_relu1[0][0]']
stage2_unit4_conv1 (Conv2D)	(None, 8, 8, 128)	147456			['zero_padding2d_14[0][0]']
stage2_unit4_bn2 (BatchNormalization)	(None, 8, 8, 128)	512			['stage2_unit4_conv1[0][0]']
stage2_unit4_relu2 (Activation)	(None, 8, 8, 128)	0			['stage2_unit4_bn2[0][0]']
)					
zero_padding2d_15 (ZeroPadding2D)	(None, 10, 10, 128)	0			['stage2_unit4_relu2[0][0]']
stage2_unit4_conv2 (Conv2D)	(None, 8, 8, 128)	147456			['zero_padding2d_15[0][0]']
add_6 (Add)	(None, 8, 8, 128)	0			['stage2_unit4_conv2[0][0]', 'add_5[0][0]']
stage3_unit1_bn1 (BatchNormalization)	(None, 8, 8, 128)	512			['add_6[0][0]']
stage3_unit1_relu1 (Activation)	(None, 8, 8, 128)	0			['stage3_unit1_bn1[0][0]']
)					

zero_padding2d_16 (ZeroPadding 2D)	(None, 10, 10, 128)	0	['stage3_unit1_relu1[0][0]']
stage3_unit1_conv1 (Conv2D)	(None, 4, 4, 256)	294912	['zero_padding2d_16[0][0]']
stage3_unit1_bn2 (BatchNormalization)	(None, 4, 4, 256)	1024	['stage3_unit1_conv1[0][0]']
stage3_unit1_relu2 (Activation)	(None, 4, 4, 256)	0	['stage3_unit1_bn2[0][0]']
zero_padding2d_17 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit1_relu2[0][0]']
stage3_unit1_conv2 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_17[0][0]']
stage3_unit1_sc (Conv2D)	(None, 4, 4, 256)	32768	['stage3_unit1_relu1[0][0]']
add_7 (Add)	(None, 4, 4, 256)	0	['stage3_unit1_conv2[0][0]', 'stage3_unit1_sc[0][0]']
stage3_unit2_bn1 (BatchNormalization)	(None, 4, 4, 256)	1024	['add_7[0][0]']
stage3_unit2_relu1 (Activation)	(None, 4, 4, 256)	0	['stage3_unit2_bn1[0][0]']
zero_padding2d_18 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit2_relu1[0][0]']
stage3_unit2_conv1 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_18[0][0]']
stage3_unit2_bn2 (BatchNormalization)	(None, 4, 4, 256)	1024	['stage3_unit2_conv1[0][0]']
stage3_unit2_relu2 (Activation)	(None, 4, 4, 256)	0	['stage3_unit2_bn2[0][0]']
zero_padding2d_19 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit2_relu2[0][0]']
stage3_unit2_conv2 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_19[0][0]']
add_8 (Add)	(None, 4, 4, 256)	0	['stage3_unit2_conv2[0][0]', 'add_7[0][0]']
stage3_unit3_bn1 (BatchNormalization)	(None, 4, 4, 256)	1024	['add_8[0][0]']
stage3_unit3_relu1 (Activation)	(None, 4, 4, 256)	0	['stage3_unit3_bn1[0][0]']
zero_padding2d_20 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit3_relu1[0][0]']
stage3_unit3_conv1 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_20[0][0]']
stage3_unit3_bn2 (BatchNormalization)	(None, 4, 4, 256)	1024	['stage3_unit3_conv1[0][0]']
stage3_unit3_relu2 (Activation)	(None, 4, 4, 256)	0	['stage3_unit3_bn2[0][0]']
zero_padding2d_21 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit3_relu2[0][0]']
stage3_unit3_conv2 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_21[0][0]']
add_9 (Add)	(None, 4, 4, 256)	0	['stage3_unit3_conv2[0][0]', 'add_8[0][0]']
stage3_unit4_bn1 (BatchNormalization)	(None, 4, 4, 256)	1024	['add_9[0][0]']
stage3_unit4_relu1 (Activation)	(None, 4, 4, 256)	0	['stage3_unit4_bn1[0][0]']

zero_padding2d_22 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit4_relu1[0][0]']
stage3_unit4_conv1 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_22[0][0]']
stage3_unit4_bn2 (BatchNormalization)	(None, 4, 4, 256)	1024	['stage3_unit4_conv1[0][0]']
stage3_unit4_relu2 (Activation)	(None, 4, 4, 256)	0	['stage3_unit4_bn2[0][0]']
zero_padding2d_23 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit4_relu2[0][0]']
stage3_unit4_conv2 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_23[0][0]']
add_10 (Add)	(None, 4, 4, 256)	0	['stage3_unit4_conv2[0][0]', 'add_9[0][0]']
stage3_unit5_bn1 (BatchNormalization)	(None, 4, 4, 256)	1024	['add_10[0][0]']
stage3_unit5_relu1 (Activation)	(None, 4, 4, 256)	0	['stage3_unit5_bn1[0][0]']
zero_padding2d_24 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit5_relu1[0][0]']
stage3_unit5_conv1 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_24[0][0]']
stage3_unit5_bn2 (BatchNormalization)	(None, 4, 4, 256)	1024	['stage3_unit5_conv1[0][0]']
stage3_unit5_relu2 (Activation)	(None, 4, 4, 256)	0	['stage3_unit5_bn2[0][0]']
zero_padding2d_25 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit5_relu2[0][0]']
stage3_unit5_conv2 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_25[0][0]']
add_11 (Add)	(None, 4, 4, 256)	0	['stage3_unit5_conv2[0][0]', 'add_10[0][0]']
stage3_unit6_bn1 (BatchNormalization)	(None, 4, 4, 256)	1024	['add_11[0][0]']
stage3_unit6_relu1 (Activation)	(None, 4, 4, 256)	0	['stage3_unit6_bn1[0][0]']
zero_padding2d_26 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit6_relu1[0][0]']
stage3_unit6_conv1 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_26[0][0]']
stage3_unit6_bn2 (BatchNormalization)	(None, 4, 4, 256)	1024	['stage3_unit6_conv1[0][0]']
stage3_unit6_relu2 (Activation)	(None, 4, 4, 256)	0	['stage3_unit6_bn2[0][0]']
zero_padding2d_27 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage3_unit6_relu2[0][0]']
stage3_unit6_conv2 (Conv2D)	(None, 4, 4, 256)	589824	['zero_padding2d_27[0][0]']
add_12 (Add)	(None, 4, 4, 256)	0	['stage3_unit6_conv2[0][0]', 'add_11[0][0]']
stage4_unit1_bn1 (BatchNormalization)	(None, 4, 4, 256)	1024	['add_12[0][0]']
stage4_unit1_relu1 (Activation)	(None, 4, 4, 256)	0	['stage4_unit1_bn1[0][0]']
zero_padding2d_28 (ZeroPadding 2D)	(None, 6, 6, 256)	0	['stage4_unit1_relu1[0][0]']

stage4_unit1_conv1 (Conv2D)	(None, 2, 2, 512)	1179648	['zero_padding2d_28[0][0]']
stage4_unit1_bn2 (BatchNormalization)	(None, 2, 2, 512)	2048	['stage4_unit1_conv1[0][0]']
stage4_unit1_relu2 (Activation)	(None, 2, 2, 512)	0	['stage4_unit1_bn2[0][0]']
zero_padding2d_29 (ZeroPadding2D)	(None, 4, 4, 512)	0	['stage4_unit1_relu2[0][0]']
stage4_unit1_conv2 (Conv2D)	(None, 2, 2, 512)	2359296	['zero_padding2d_29[0][0]']
stage4_unit1_sc (Conv2D)	(None, 2, 2, 512)	131072	['stage4_unit1_relu1[0][0]']
add_13 (Add)	(None, 2, 2, 512)	0	['stage4_unit1_conv2[0][0]', 'stage4_unit1_sc[0][0]']
stage4_unit2_bn1 (BatchNormalization)	(None, 2, 2, 512)	2048	['add_13[0][0]']
stage4_unit2_relu1 (Activation)	(None, 2, 2, 512)	0	['stage4_unit2_bn1[0][0]']
zero_padding2d_30 (ZeroPadding2D)	(None, 4, 4, 512)	0	['stage4_unit2_relu1[0][0]']
stage4_unit2_conv1 (Conv2D)	(None, 2, 2, 512)	2359296	['zero_padding2d_30[0][0]']
stage4_unit2_bn2 (BatchNormalization)	(None, 2, 2, 512)	2048	['stage4_unit2_conv1[0][0]']
stage4_unit2_relu2 (Activation)	(None, 2, 2, 512)	0	['stage4_unit2_bn2[0][0]']
zero_padding2d_31 (ZeroPadding2D)	(None, 4, 4, 512)	0	['stage4_unit2_relu2[0][0]']
stage4_unit2_conv2 (Conv2D)	(None, 2, 2, 512)	2359296	['zero_padding2d_31[0][0]']
add_14 (Add)	(None, 2, 2, 512)	0	['stage4_unit2_conv2[0][0]', 'add_13[0][0]']
stage4_unit3_bn1 (BatchNormalization)	(None, 2, 2, 512)	2048	['add_14[0][0]']
stage4_unit3_relu1 (Activation)	(None, 2, 2, 512)	0	['stage4_unit3_bn1[0][0]']
zero_padding2d_32 (ZeroPadding2D)	(None, 4, 4, 512)	0	['stage4_unit3_relu1[0][0]']
stage4_unit3_conv1 (Conv2D)	(None, 2, 2, 512)	2359296	['zero_padding2d_32[0][0]']
stage4_unit3_bn2 (BatchNormalization)	(None, 2, 2, 512)	2048	['stage4_unit3_conv1[0][0]']
stage4_unit3_relu2 (Activation)	(None, 2, 2, 512)	0	['stage4_unit3_bn2[0][0]']
zero_padding2d_33 (ZeroPadding2D)	(None, 4, 4, 512)	0	['stage4_unit3_relu2[0][0]']
stage4_unit3_conv2 (Conv2D)	(None, 2, 2, 512)	2359296	['zero_padding2d_33[0][0]']
add_15 (Add)	(None, 2, 2, 512)	0	['stage4_unit3_conv2[0][0]', 'add_14[0][0]']
bn1 (BatchNormalization)	(None, 2, 2, 512)	2048	['add_15[0][0]']
relu1 (Activation)	(None, 2, 2, 512)	0	['bn1[0][0]']
fpn_stage_p5_pre_conv (Conv2D)	(None, 2, 2, 64)	32832	['relu1[0][0]']
fpn_stage_p5_upsampling (UpSampling2D)	(None, 4, 4, 64)	0	['fpn_stage_p5_pre_conv[0][0]']

fpn_stage_p5_conv (Conv2D)	(None, 4, 4, 64)	16448	['stage4_unit1_relu[0][0]']
fpn_stage_p5_add (Add)	(None, 4, 4, 64)	0	['fpn_stage_p5_upsampling[0][0]', 'fpn_stage_p5_conv[0][0]']
fpn_stage_p4_upsampling (UpSampling2D)	(None, 8, 8, 64)	0	['fpn_stage_p5_add[0][0]']
fpn_stage_p4_conv (Conv2D)	(None, 8, 8, 64)	8256	['stage3_unit1_relu[0][0]']
fpn_stage_p4_add (Add)	(None, 8, 8, 64)	0	['fpn_stage_p4_upsampling[0][0]', 'fpn_stage_p4_conv[0][0]']
fpn_stage_p3_upsampling (UpSampling2D)	(None, 16, 16, 64)	0	['fpn_stage_p4_add[0][0]']
fpn_stage_p3_conv (Conv2D)	(None, 16, 16, 64)	4160	['stage2_unit1_relu[0][0]']
fpn_stage_p3_add (Add)	(None, 16, 16, 64)	0	['fpn_stage_p3_upsampling[0][0]', 'fpn_stage_p3_conv[0][0]']
fpn_stage_p2_upsampling (UpSampling2D)	(None, 32, 32, 64)	0	['fpn_stage_p3_add[0][0]']
fpn_stage_p2_conv (Conv2D)	(None, 32, 32, 64)	4160	['relu0[0][0]']
fpn_stage_p2_add (Add)	(None, 32, 32, 64)	0	['fpn_stage_p2_upsampling[0][0]', 'fpn_stage_p2_conv[0][0]']
segm_stage3a_conv (Conv2D)	(None, 16, 16, 32)	18432	['fpn_stage_p3_add[0][0]']
segm_stage4a_conv (Conv2D)	(None, 8, 8, 32)	18432	['fpn_stage_p4_add[0][0]']
segm_stage5a_conv (Conv2D)	(None, 4, 4, 32)	18432	['fpn_stage_p5_add[0][0]']
segm_stage2a_conv (Conv2D)	(None, 32, 32, 32)	18432	['fpn_stage_p2_add[0][0]']
segm_stage3a_bn (BatchNormalization)	(None, 16, 16, 32)	128	['segm_stage3a_conv[0][0]']
segm_stage4a_bn (BatchNormalization)	(None, 8, 8, 32)	128	['segm_stage4a_conv[0][0]']
segm_stage5a_bn (BatchNormalization)	(None, 4, 4, 32)	128	['segm_stage5a_conv[0][0]']
segm_stage2a_bn (BatchNormalization)	(None, 32, 32, 32)	128	['segm_stage2a_conv[0][0]']
segm_stage3a_relu (Activation)	(None, 16, 16, 32)	0	['segm_stage3a_bn[0][0]']
segm_stage4a_relu (Activation)	(None, 8, 8, 32)	0	['segm_stage4a_bn[0][0]']
segm_stage5a_relu (Activation)	(None, 4, 4, 32)	0	['segm_stage5a_bn[0][0]']
segm_stage2a_relu (Activation)	(None, 32, 32, 32)	0	['segm_stage2a_bn[0][0]']
segm_stage3b_conv (Conv2D)	(None, 16, 16, 32)	9216	['segm_stage3a_relu[0][0]']
segm_stage4b_conv (Conv2D)	(None, 8, 8, 32)	9216	['segm_stage4a_relu[0][0]']
segm_stage5b_conv (Conv2D)	(None, 4, 4, 32)	9216	['segm_stage5a_relu[0][0]']
segm_stage2b_conv (Conv2D)	(None, 32, 32, 32)	9216	['segm_stage2a_relu[0][0]']
segm_stage3b_bn (BatchNormalization)	(None, 16, 16, 32)	128	['segm_stage3b_conv[0][0]']
segm_stage4b_bn (BatchNormalization)	(None, 8, 8, 32)	128	['segm_stage4b_conv[0][0]']
segm_stage5b_bn (BatchNormalization)	(None, 4, 4, 32)	128	['segm_stage5b_conv[0][0]']
segm_stage2b_bn (BatchNormalization)	(None, 32, 32, 32)	128	['segm_stage2b_conv[0][0]']

segm_stage3b_relu (Activation)	(None, 16, 16, 32)	0	['segm_stage3b_bn[0][0]']
segm_stage4b_relu (Activation)	(None, 8, 8, 32)	0	['segm_stage4b_bn[0][0]']
segm_stage5b_relu (Activation)	(None, 4, 4, 32)	0	['segm_stage5b_bn[0][0]']
segm_stage2b_relu (Activation)	(None, 32, 32, 32)	0	['segm_stage2b_bn[0][0]']
upsampling_stage3 (UpSampling2D)	(None, 32, 32, 32)	0	['segm_stage3b_relu[0][0]']
upsampling_stage4 (UpSampling2D)	(None, 32, 32, 32)	0	['segm_stage4b_relu[0][0]']
upsampling_stage5 (UpSampling2D)	(None, 32, 32, 32)	0	['segm_stage5b_relu[0][0]']
aggregation_concat (Concatenate)	(None, 32, 32, 128)	0	['segm_stage2b_relu[0][0]', 'upsampling_stage3[0][0]', 'upsampling_stage4[0][0]', 'upsampling_stage5[0][0]']
final_stage_conv (Conv2D)	(None, 32, 32, 32)	36864	['aggregation_concat[0][0]']
final_stage_bn (BatchNormalization)	(None, 32, 32, 32)	128	['final_stage_conv[0][0]']
final_stage_relu (Activation)	(None, 32, 32, 32)	0	['final_stage_bn[0][0]']
final_upsampling (UpSampling2D)	(None, 64, 64, 32)	0	['final_stage_relu[0][0]']
head_conv (Conv2D)	(None, 64, 64, 17)	4913	['final_upsampling[0][0]']
softmax (Activation)	(None, 64, 64, 17)	0	['head_conv[0][0]']

Total params: 21,810,638
Trainable params: 21,794,512
Non-trainable params: 16,126

In []:

```
# del FPN_base_model
```

FPN with Attention Mechanism

In []:

```
class Attention_LAM(tensorflow.keras.layers.Layer):
    # This class implements Linear Attention Mechanism described in the 'A2-FPN for Semantic Segmentation of
    # Fine-Resolution Remotely Sensed Images' (arXiv:2102.07997v3)

    def __init__(self, k=64 , scale = 8 ):
        super(Attention,self).__init__()
        self.eps = 1e-6
        self.qkrnls = int(k/scale)
        self.vkrnls = k
        self.gamma = tf.Variable(0.0)
        self.Q_conv = Conv2D(self.qkrnls,(1,1))
        self.K_conv = Conv2D(self.qkrnls,(1,1))
        self.V_conv = Conv2D(self.vkrnls,(1,1))

    def call(self,X):
        batch,width,height,chnl = X.shape
        # print(batch,width*height,chnl)
        N = int(width*height)

        Q = tf.reshape(self.Q_conv(X),[-1,N,self.qkrnls]) # -1 represents None ~ Shape not known
        K = tf.reshape(self.K_conv(X),[-1,N,self.qkrnls])
```

```

V = tf.reshape(self.V_conv(X), [-1,N,self.vkrnls])

# print(Q.shape,K.shape,V.shape)

Q = tf.einsum('bnc, bn->bnc', Q, 1/tf.norm(Q, ord='euclidean', axis=-1))
K = tf.einsum('bnc, bn->bnc', K, 1/tf.norm(K, ord='euclidean', axis=-1))

K = tf.transpose(K, [0,2,1])
V = tf.transpose(V, [0,2,1])

# print(K.shape,V.shape)

Deno = 1 / (width*height + tf.einsum('bnc, bc->bn', Q, tf.reduce_sum(K,-1)) + self.eps)
# print('Deno', Deno.shape)

V_sum = tf.tile(tf.reduce_sum(V,-1,keepdims=True), [1,1,width*height])
KV = tf.einsum('bmn, bcn->bmc', K, V)

# print('V_sum', V_sum.shape, 'KV', KV.shape)

Nume = V_sum + tf.einsum('bnm, bmc->bcn', Q, KV)

D = tf.einsum('bcn, bn->bcn', Nume, Deno)
# print('D', D.shape)

D = tf.reshape(D, [-1,D.shape[1],width,height])
# print('D', D.shape)

D = tf.transpose(D, [0,2,3,1])

# print('Nume', Nume.shape, 'D', D.shape)

return self.gamma * D

```

Note : It was observed that the LAM had no impact on training time when compared to Dot product mechanism. Hence the Dot product Attention Mechanism is considered for Model implementation

In [42]:

```

class Attention(tensorflow.keras.layers.Layer):
    # This class implements Dot-Product Attention Mechanism described in the 'A2-FPN for Semantic Segmentation of
    # Fine-Resolution Remotely Sensed Images' (arXiv:2102.07997v3)

    def __init__(self, k=256 , scale = 8 ):
        super(Attention, self) . __init__()
        self.eps = 1e-6
        self.qkrnls = int(k/scale)
        self.vkrnls = k
        self.gamma = tf.Variable(0.0)
        self.Q_conv = Conv2D(self.qkrnls,(1,1))
        self.K_conv = Conv2D(self.qkrnls,(1,1))
        self.V_conv = Conv2D(self.vkrnls,(1,1))

    def call(self,X):
        batch, width, height, chnl = X.shape
        # print(batch, width*height, chnl)
        N = int(width*height)

        Q = tf.reshape(self.Q_conv(X), [-1,N,self.qkrnls]) # -1 represents Shape not known
        K = tf.reshape(self.K_conv(X), [-1,N,self.qkrnls])
        V = tf.reshape(self.V_conv(X), [-1,N,self.vkrnls])

        # print(Q.shape,K.shape,V.shape)

        KT = tf.transpose(K, [0,2,1])

        # print(K.shape,V.shape)

        QKT = tf.matmul(Q, KT)
        # print('QK', QKT.shape)

        Sf_QKT = tf.keras.activations.softmax(QKT)
        # print('SF_QK', Sf_QKT.shape)

```

```

D = tf.matmul(Sf_QKT,V)

# print('D',D.shape)

D = tf.reshape(D, [-1, width, height, D.shape[2]])
# print('D',D.shape)

return self.gamma * D

```

In [43]:

```

class A2_Block(tensorflow.keras.layers.Layer):
    # This class implements Attention Module described in the 'A2-FPN for Semantic Segmentation of
    # Fine-Resolution Remotely Sensed Images' (arXiv:2102.07997v3)

    def __init__(self,f=256):
        super().__init__(name='A2_Block')
        self.aconv = Conv2D(filters = f, kernel_size = (1,1), padding='valid')
        self.attention = Attention(k=256)

    def call(self,X):#s1,s2,s3,s4):
        # X = Concatenate(axis=-1) ([s1,s2,s3,s4])
        X_conv = self.aconv(X)
        # print('conv before attn',X_conv.shape)
        X_att = self.attention(X_conv) #LAM
        # print('conv after attn',X_att.shape)
        X_out = X_conv + X_att
        return X_out

```

In [44]:

```

class A2FPN_Model(tensorflow.keras.Model):
    # This class implements FPN based Model with Attention described in the 'A2-FPN for Semantic Segmentation of
    # Fine-Resolution Remotely Sensed Images (arXiv:2102.07997v3)
    def __init__(self) :
        super().__init__(name='A2FPN')

        FPN_base_model = FPN(backbone_name='resnet34',
                             input_shape=(64, 64, 95),
                             classes=17,
                             activation='softmax',
                             weights=None, encoder_weights=None, #'imagenet',
                             encoder_freeze=False,
                             encoder_features='default',
                             pyramid_block_filters=128,
                             pyramid_use_batchnorm=True,
                             pyramid_aggregation='concat', pyramid_dropout=None)

        # FPN_res = FPN_base_model.layers[-7]
        self.FPN_base_model_mod = Model(inputs = FPN_base_model.input, outputs = FPN_base_model.layers[-7].output)

        self.attention = A2_Block(f=256)

        self.dropout = Dropout(0.2)

        self.up = UpSampling2D(size=(2, 2),interpolation='bilinear')

        self.final_conv = Conv2D(17, (1, 1), name='conv_final', activation = 'softmax', padding="valid",
                               kernel_initializer='glorot_uniform')

    def call(self,X):

        seg_concat = self.FPN_base_model_mod(X)

        att = self.attention(seg_concat)
        out = self.dropout(att)
        out = self.up(out)
        out = self.final_conv(out)

        return out

```

```
#https://stackoverflow.com/questions/61427583/how-do-i-plot-a-keras-tensorflow-subclassing-api-model
```

```
def build_model(self):
    x = Input(shape=(64, 64, 95))
    return Model(inputs=[x], outputs=self.call(x), name='A2FPN_resnet')
```

In [45]:

```
# del model_obj,FPN_model
```

In [46]:

```
# Model Object Definition
model_obj = A2FPN_Model()
FPN_model = model_obj.build_model()
```

In [47]:

```
FPN_model.summary()
```

Model: "A2FPN_resnet"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 64, 64, 95)]	0
model_2 (Functional)	(None, 32, 32, 256)	22167389
A2_Block (A2_Block)	(None, 32, 32, 256)	148033
dropout (Dropout)	(None, 32, 32, 256)	0
up_sampling2d (UpSampling2D	(None, 64, 64, 256)	0
)		
conv_final (Conv2D)	(None, 64, 64, 17)	4369
<hr/>		
Total params: 22,319,791		
Trainable params: 22,303,217		
Non-trainable params: 16,574		

Model compile

Dice loss

Formulation:

Dice loss is based on dice coefficient.

Dice coefficient captures the amount of overlap between two sets. It is ratio of intersection of two sets to union of two sets.

Dice coefficient is given by :

$$\text{Dice coefficient} = 2 \cdot \frac{|X \cap Y|}{|X| + |Y|}$$

When we apply for evaluating loss, X will be ground truth while Y will be predictions. The loss is given by:

$$\text{Dice loss} = 1 - \text{Dice coefficient}$$

The intersection is approximated as dot product of ground truth and predictions and in denominator predictions and ground truths are summed up

$$\text{Dice loss} = 1 - \frac{2 \cdot \sum y_g \cdot \sum y_p}{\sum y_g + \sum y_p}$$

y_g = ground truth, y_p = prediction

Dice loss can also be expressed in terms of F1 score:

$$\text{Dice loss} = 1 - \text{F}_1$$

General Fscore : $F_{\beta} = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$

when $\beta = 1$, we have F1 score given by

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
 $\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$

Replacing F1 score with above expression for precision and recall

$$F_1 = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

$$\text{Dice loss} = 1 - \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

Range of loss function:

We have F1 score range in [0,1] Thus we will have the dice loss in same range [0,1]

When F1 score is 0, Dice loss is 1

When F1 score is 1, Dice loss is 0

Interpretation of loss function:

Loss function will be high ($F_1=0$) when there are NO True positive predictions by the model

Loss function will be low ($F_1=1$), when there All actual positive are predicted as True positives and no False positive predictions by the model. i.e. Both Precision and Recall of the obtained results are high

Loss for segmentation problem:

In segmentation problem, we have array of masked images generated for category of classes for every image to train the model.

Masked image of particular class have value 1 in the array location of identified class of object in the image, while other locations in array are marked 0.

Predicted images by the Unet model will also be of arrays of 0s and 1s stacked together where stack size is number of classes.

To evaluate the loss, F1 score of predicted classes and actual classes are evaluated.

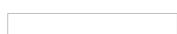
When Predicted region is not exactly same as the actual location of object in the image, there will be regions of image which will be False positive and False negative. Thus reducing precision and recall which affects F1 score. Thus increases the loss.

When the model is trained to minimize the loss. the model will predict exact location of the objects.

Consider a image shown below. Let Blue region be a object identified and masked as 1s and Red region be the prediction from model.

Now the region where prediction and the actual mask overlap is the region where model has correctly predicted. Blue region which are not under Red region are False negatives and Red region without overlap are False positives.

Thus we can see that when TP counts are low as the model has not predicted the object correctly. Hence loss will be close to 1. As the model gets trained, False negatives and False positives decrease. This improves the F1 score and loss decreases



https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient

https://github.com/qubvel/segmentation_models/blob/94f624b7029deb463c859efbd92fa26f512b52b8/segmentation_models/losses.py

<https://en.wikipedia.org/wiki/F-score>

Model Training

In []:

```
optim = tf.keras.optimizers.Adam(0.0001)

focal_loss = sm.losses.cce_dice_loss #cce_dice_loss = categorical_crossentropy + dice_loss

FPN_model.compile(optim, focal_loss, metrics=[iou_score])
```

20220526-153302 WARNING:tensorflow:write_grads will be ignored in TensorFlow 2.0 for the TensorBoard Callback.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:21: UserWarning: Model.fit_generator is deprecated and will be removed in a future version. Please use Model.fit, which supports generators. WARNING:tensorflow:Model failed to serialize as JSON. Ignoring... Layer A2_Block has arguments ['self', 'f'] in __init__ and therefore must override get_config().

Example:

```
class CustomLayer(keras.layers.Layer): def __init__(self, arg1, arg2): super().__init__() self.arg1 = arg1 self.arg2 = arg2

    def get_config(self):
        config = super().get_config()
        config.update({
            "arg1": self.arg1,
            "arg2": self.arg2,
        })
        return config
```

Epoch 1/50 80/80 [=====] - ETA: 0s - loss: 0.9688 - iou_score: 0.0858 Epoch 1: saving model to model_2_e50/FPN_best_model_e01.h5 80/80 [=====] - 30s 154ms/step - loss: 0.9688 - iou_score: 0.0858 - val_loss: 1.5670 - val_iou_score: 0.0342 - lr: 1.0000e-04 Epoch 2/50 80/80 [=====] - ETA: 0s - loss: 0.7628 - iou_score: 0.2089 Epoch 2: saving model to model_2_e50/FPN_best_model_e02.h5 80/80 [=====] - 11s 138ms/step - loss: 0.7628 - iou_score: 0.2089 - val_loss: 1.0004 - val_iou_score: 0.1098 - lr: 1.0000e-04 Epoch 3/50 80/80 [=====] - ETA: 0s - loss: 0.5944 - iou_score: 0.3374 Epoch 3: saving model to model_2_e50/FPN_best_model_e03.h5 80/80 [=====] - 11s 140ms/step - loss: 0.5944 - iou_score: 0.3374 - val_loss: 0.6935 - val_iou_score: 0.2695 - lr: 1.0000e-04 Epoch 4/50 80/80 [=====] - ETA: 0s - loss: 0.4643 - iou_score: 0.4541 Epoch 4: saving model to model_2_e50/FPN_best_model_e04.h5 80/80 [=====] - 11s 139ms/step - loss: 0.4643 - iou_score: 0.4541 - val_loss: 0.4487 - val_iou_score: 0.4664 - lr: 1.0000e-04 Epoch 5/50 80/80 [=====] - ETA: 0s - loss: 0.3980 - iou_score: 0.5206 Epoch 5: saving model to model_2_e50/FPN_best_model_e05.h5 80/80 [=====] - 11s 139ms/step - loss: 0.3980 - iou_score: 0.5206 - val_loss: 0.4005 - val_iou_score: 0.5128 - lr: 1.0000e-04 Epoch 6/50 80/80 [=====] - ETA: 0s - loss: 0.3488 - iou_score: 0.5679 Epoch 6: saving model to model_2_e50/FPN_best_model_e06.h5

Epoch 6: ReduceLROnPlateau reducing learning rate to 8.99999772640876e-05. 80/80 [=====] - 11s 141ms/step - loss: 0.3488 - iou_score: 0.5679 - val_loss: 0.3639 - val_iou_score: 0.5474 - lr: 1.0000e-04 Epoch 7/50 80/80 [=====] - ETA: 0s - loss: 0.3105 - iou_score: 0.6056 Epoch 7: saving model to model_2_e50/FPN_best_model_e07.h5 80/80 [=====] - 11s 140ms/step - loss: 0.3105 - iou_score: 0.6056 - val_loss: 0.3181 - val_iou_score: 0.5893 - lr: 9.0000e-05 Epoch 8/50 80/80 [=====] - ETA: 0s - loss: 0.2678 - iou_score: 0.6471 Epoch 8: saving model to model_2_e50/FPN_best_model_e08.h5 80/80 [=====] - 12s 147ms/step - loss: 0.2678 - iou_score: 0.6471 - val_loss: 0.2765 - val_iou_score: 0.6324 - lr: 9.0000e-05 Epoch 9/50 80/80 [=====] - ETA: 0s - loss: 0.2370 - iou_score: 0.6812 Epoch 9: saving model to model_2_e50/FPN_best_model_e09.h5 80/80 [=====] - 11s 137ms/step - loss: 0.2370 - iou_score: 0.6812 - val_loss: 0.2521 - val_iou_score: 0.6579 - lr: 9.0000e-05 Epoch 10/50 80/80 [=====] - ETA: 0s - loss: 0.2151 - iou_score: 0.7086 Epoch 10: saving model to model_2_e50/FPN_best_model_e10.h5 80/80 [=====] - 11s 139ms/step - loss: 0.2151 - iou_score: 0.7086 - val_loss: 0.2268 - val_iou_score: 0.6890 - lr: 9.0000e-05 Epoch 11/50 80/80 [=====] - ETA: 0s - loss: 0.2005 - iou_score: 0.7270 Epoch 11: saving model to model_2_e50/FPN_best_model_e11.h5

Epoch 11: ReduceLROnPlateau reducing learning rate to 8.100000122794882e-05. 80/80 [=====] - 11s 138ms/step - loss: 0.2005 - iou_score: 0.7270 - val_loss: 0.2204 - val_iou_score: 0.6968 - lr: 9.0000e-05 Epoch 12/50 80/80 [=====] - ETA: 0s - loss: 0.1873 - iou_score: 0.7435 Epoch 12: saving model to model_2_e50/FPN_best_model_e12.h5 80/80 [=====] - 11s 138ms/step - loss: 0.1873 - iou_score: 0.7435 - val_loss: 0.2152 - val_iou_score: 0.7048 - lr: 8.1000e-05 Epoch 13/50 80/80 [=====] - ETA: 0s - loss: 0.1789 - iou_score: 0.7548 Epoch 13: saving model to model_2_e50/FPN_best_model_e13.h5 80/80 [=====] - 11s 140ms/step - loss: 0.1789 - iou_score: 0.7548 - val_loss: 0.1963 - val_iou_score: 0.7278 - lr: 8.1000e-05 Epoch 14/50 80/80 [=====] - ETA: 0s - loss: 0.1716 - iou_score: 0.7652

0.7270 - II. 8.1000e-05 Epoch 14/50 80/80 [=====] - ETA: 0s - loss: 0.1716 - iou_score: 0.7652
Epoch 14: saving model to model_2_e50/FPN_best_model_e14.h5 80/80 [=====] - 11s 139ms/step
- loss: 0.1716 - iou_score: 0.7652 - val_loss: 0.1887 - val_iou_score: 0.7370 - lr: 8.1000e-05 Epoch 15/50 80/80
[=====] - ETA: 0s - loss: 0.1649 - iou_score: 0.7743 Epoch 15: saving model to
model_2_e50/FPN_best_model_e15.h5 80/80 [=====] - 11s 139ms/step - loss: 0.1649 - iou_score:
0.7743 - val_loss: 0.1843 - val_iou_score: 0.7433 - lr: 8.1000e-05 Epoch 16/50 80/80 [=====]
ETA: 0s - loss: 0.1592 - iou_score: 0.7819 Epoch 16: saving model to model_2_e50/FPN_best_model_e16.h5

Epoch 16: ReduceLROnPlateau reducing learning rate to 7.289999848580919e-05. 80/80 [=====]
- 11s 139ms/step - loss: 0.1592 - iou_score: 0.7819 - val_loss: 0.1832 - val_iou_score: 0.7447 - lr: 8.1000e-05 Epoch 17/50 80/80
[=====] - ETA: 0s - loss: 0.1531 - iou_score: 0.7906 Epoch 17: saving model to
model_2_e50/FPN_best_model_e17.h5 80/80 [=====] - 11s 138ms/step - loss: 0.1531 - iou_score:
0.7906 - val_loss: 0.1817 - val_iou_score: 0.7466 - lr: 7.2900e-05 Epoch 18/50 80/80 [=====]
ETA: 0s - loss: 0.1477 - iou_score: 0.7981 Epoch 18: saving model to model_2_e50/FPN_best_model_e18.h5 80/80
[=====] - 12s 145ms/step - loss: 0.1477 - iou_score: 0.7981 - val_loss: 0.1659 - val_iou_score:
0.7675 - lr: 7.2900e-05 Epoch 19/50 80/80 [=====] - ETA: 0s - loss: 0.1465 - iou_score: 0.8000
Epoch 19: saving model to model_2_e50/FPN_best_model_e19.h5 80/80 [=====] - 11s 140ms/step
- loss: 0.1465 - iou_score: 0.8000 - val_loss: 0.1662 - val_iou_score: 0.7667 - lr: 7.2900e-05 Epoch 20/50 80/80
[=====] - ETA: 0s - loss: 0.1419 - iou_score: 0.8062 Epoch 20: saving model to
model_2_e50/FPN_best_model_e20.h5 80/80 [=====] - 11s 139ms/step - loss: 0.1419 - iou_score:
0.8062 - val_loss: 0.1796 - val_iou_score: 0.7503 - lr: 7.2900e-05 Epoch 21/50 80/80 [=====]
ETA: 0s - loss: 0.1407 - iou_score: 0.8082 Epoch 21: saving model to model_2_e50/FPN_best_model_e21.h5

Epoch 21: ReduceLROnPlateau reducing learning rate to 6.56100019114092e-05. 80/80 [=====]
- 11s 139ms/step - loss: 0.1407 - iou_score: 0.8082 - val_loss: 0.1612 - val_iou_score: 0.7738 - lr: 7.2900e-05 Epoch 22/50 80/80
[=====] - ETA: 0s - loss: 0.1347 - iou_score: 0.8167 Epoch 22: saving model to
model_2_e50/FPN_best_model_e22.h5 80/80 [=====] - 11s 140ms/step - loss: 0.1347 - iou_score:
0.8167 - val_loss: 0.1626 - val_iou_score: 0.7729 - lr: 6.5610e-05 Epoch 23/50 80/80 [=====]
ETA: 0s - loss: 0.1312 - iou_score: 0.8217 Epoch 23: saving model to model_2_e50/FPN_best_model_e23.h5 80/80
[=====] - 11s 138ms/step - loss: 0.1312 - iou_score: 0.8217 - val_loss: 0.1536 - val_iou_score:
0.7849 - lr: 6.5610e-05 Epoch 24/50 80/80 [=====] - ETA: 0s - loss: 0.1257 - iou_score: 0.8287
Epoch 24: saving model to model_2_e50/FPN_best_model_e24.h5 80/80 [=====] - 11s 139ms/step
- loss: 0.1257 - iou_score: 0.8287 - val_loss: 0.1551 - val_iou_score: 0.7827 - lr: 6.5610e-05 Epoch 25/50 80/80
[=====] - ETA: 0s - loss: 0.1254 - iou_score: 0.8296 Epoch 25: saving model to
model_2_e50/FPN_best_model_e25.h5 80/80 [=====] - 11s 139ms/step - loss: 0.1254 - iou_score:
0.8296 - val_loss: 0.1524 - val_iou_score: 0.7860 - lr: 6.5610e-05 Epoch 26/50 80/80 [=====]
ETA: 0s - loss: 0.1242 - iou_score: 0.8315 Epoch 26: saving model to model_2_e50/FPN_best_model_e26.h5

Epoch 26: ReduceLROnPlateau reducing learning rate to 5.904900172026828e-05. 80/80 [=====]
- 11s 141ms/step - loss: 0.1242 - iou_score: 0.8315 - val_loss: 0.1661 - val_iou_score: 0.7687 - lr: 6.5610e-05 Epoch 27/50 80/80
[=====] - ETA: 0s - loss: 0.1259 - iou_score: 0.8289 Epoch 27: saving model to
model_2_e50/FPN_best_model_e27.h5 80/80 [=====] - 11s 140ms/step - loss: 0.1259 - iou_score:
0.8289 - val_loss: 0.1456 - val_iou_score: 0.7957 - lr: 5.9049e-05 Epoch 28/50 80/80 [=====]
ETA: 0s - loss: 0.1188 - iou_score: 0.8393 Epoch 28: saving model to model_2_e50/FPN_best_model_e28.h5 80/80
[=====] - 12s 145ms/step - loss: 0.1188 - iou_score: 0.8393 - val_loss: 0.1457 - val_iou_score:
0.7962 - lr: 5.9049e-05 Epoch 29/50 80/80 [=====] - ETA: 0s - loss: 0.1150 - iou_score: 0.8445
Epoch 29: saving model to model_2_e50/FPN_best_model_e29.h5 80/80 [=====] - 11s 140ms/step
- loss: 0.1150 - iou_score: 0.8445 - val_loss: 0.1419 - val_iou_score: 0.8000 - lr: 5.9049e-05 Epoch 30/50 80/80
[=====] - ETA: 0s - loss: 0.1125 - iou_score: 0.8484 Epoch 30: saving model to
model_2_e50/FPN_best_model_e30.h5 80/80 [=====] - 11s 142ms/step - loss: 0.1125 - iou_score:
0.8484 - val_loss: 0.1445 - val_iou_score: 0.7974 - lr: 5.9049e-05 Epoch 31/50 80/80 [=====]
ETA: 0s - loss: 0.1116 - iou_score: 0.8496 Epoch 31: saving model to model_2_e50/FPN_best_model_e31.h5

Epoch 31: ReduceLROnPlateau reducing learning rate to 5.314410154824145e-05. 80/80 [=====]
- 11s 142ms/step - loss: 0.1116 - iou_score: 0.8496 - val_loss: 0.1399 - val_iou_score: 0.8039 - lr: 5.9049e-05 Epoch 32/50 80/80
[=====] - ETA: 0s - loss: 0.1079 - iou_score: 0.8551 Epoch 32: saving model to
model_2_e50/FPN_best_model_e32.h5 80/80 [=====] - 11s 141ms/step - loss: 0.1079 - iou_score:
0.8551 - val_loss: 0.1393 - val_iou_score: 0.8054 - lr: 5.3144e-05 Epoch 33/50 80/80 [=====]
ETA: 0s - loss: 0.1056 - iou_score: 0.8579 Epoch 33: saving model to model_2_e50/FPN_best_model_e33.h5 80/80
[=====] - 11s 141ms/step - loss: 0.1056 - iou_score: 0.8579 - val_loss: 0.1388 - val_iou_score:
0.8059 - lr: 5.3144e-05 Epoch 34/50 80/80 [=====] - ETA: 0s - loss: 0.1092 - iou_score: 0.8535
Epoch 34: saving model to model_2_e50/FPN_best_model_e34.h5 80/80 [=====] - 11s 141ms/step
- loss: 0.1092 - iou_score: 0.8535 - val_loss: 0.1381 - val_iou_score: 0.8062 - lr: 5.3144e-05 Epoch 35/50 80/80
[=====] - ETA: 0s - loss: 0.1081 - iou_score: 0.8549 Epoch 35: saving model to
model_2_e50/FPN_best_model_e35.h5 80/80 [=====] - 11s 142ms/step - loss: 0.1081 - iou_score:
0.8549 - val_loss: 0.1354 - val_iou_score: 0.8102 - lr: 5.3144e-05 Epoch 36/50 80/80 [=====]
ETA: 0s - loss: 0.1035 - iou_score: 0.8610 Epoch 36: saving model to model_2_e50/FPN_best_model_e36.h5

Epoch 36: ReduceLROnPlateau reducing learning rate to 4.7829690083744934e-05. 80/80
[=====] - 11s 140ms/step - loss: 0.1035 - iou_score: 0.8610 - val_loss: 0.1376 - val_iou_score: 0.8081 - lr: 5.3144e-05 Epoch 37/50 80/80 [=====] - ETA: 0s - loss: 0.1012 - iou_score: 0.8643
Epoch 37: saving model to model_2_e50/FPN_best_model_e37.h5 80/80 [=====] - 11s 140ms/step - loss: 0.1012 - iou_score: 0.8643 - val_loss: 0.1333 - val_iou_score: 0.8141 - lr: 4.7830e-05 Epoch 38/50 80/80
[=====] - ETA: 0s - loss: 0.1001 - iou_score: 0.8662 Epoch 38: saving model to model_2_e50/FPN_best_model_e38.h5 80/80 [=====] - 11s 139ms/step - loss: 0.1001 - iou_score: 0.8662 - val_loss: 0.1343 - val_iou_score: 0.8125 - lr: 4.7830e-05 Epoch 39/50 80/80 [=====] - ETA: 0s - loss: 0.0982 - iou_score: 0.8689 Epoch 39: saving model to model_2_e50/FPN_best_model_e39.h5 80/80
[=====] - 11s 141ms/step - loss: 0.0982 - iou_score: 0.8689 - val_loss: 0.1346 - val_iou_score: 0.8114 - lr: 4.7830e-05 Epoch 40/50 80/80 [=====] - ETA: 0s - loss: 0.0991 - iou_score: 0.8684
Epoch 40: saving model to model_2_e50/FPN_best_model_e40.h5 80/80 [=====] - 11s 141ms/step - loss: 0.0991 - iou_score: 0.8684 - val_loss: 0.1309 - val_iou_score: 0.8174 - lr: 4.7830e-05 Epoch 41/50 80/80
[=====] - ETA: 0s - loss: 0.0967 - iou_score: 0.8715 Epoch 41: saving model to model_2_e50/FPN_best_model_e41.h5

Epoch 41: ReduceLROnPlateau reducing learning rate to 4.304672074795235e-05. 80/80 [=====]
- 11s 140ms/step - loss: 0.0967 - iou_score: 0.8715 - val_loss: 0.1372 - val_iou_score: 0.8090 - lr: 4.7830e-05 Epoch 42/50 80/80
[=====] - ETA: 0s - loss: 0.0968 - iou_score: 0.8710 Epoch 42: saving model to model_2_e50/FPN_best_model_e42.h5 80/80 [=====] - 11s 140ms/step - loss: 0.0968 - iou_score: 0.8710 - val_loss: 0.1335 - val_iou_score: 0.8142 - lr: 4.3047e-05 Epoch 43/50 80/80 [=====] - ETA: 0s - loss: 0.0943 - iou_score: 0.8743 Epoch 43: saving model to model_2_e50/FPN_best_model_e43.h5 80/80
[=====] - 11s 140ms/step - loss: 0.0943 - iou_score: 0.8743 - val_loss: 0.1336 - val_iou_score: 0.8139 - lr: 4.3047e-05 Epoch 44/50 80/80 [=====] - ETA: 0s - loss: 0.0945 - iou_score: 0.8745
Epoch 44: saving model to model_2_e50/FPN_best_model_e44.h5 80/80 [=====] - 11s 141ms/step - loss: 0.0945 - iou_score: 0.8745 - val_loss: 0.1312 - val_iou_score: 0.8176 - lr: 4.3047e-05 Epoch 45/50 80/80
[=====] - ETA: 0s - loss: 0.0926 - iou_score: 0.8770 Epoch 45: saving model to model_2_e50/FPN_best_model_e45.h5 80/80 [=====] - 11s 139ms/step - loss: 0.0926 - iou_score: 0.8770 - val_loss: 0.1295 - val_iou_score: 0.8191 - lr: 4.3047e-05 Epoch 46/50 80/80 [=====] - ETA: 0s - loss: 0.0909 - iou_score: 0.8790 Epoch 46: saving model to model_2_e50/FPN_best_model_e46.h5

Epoch 46: ReduceLROnPlateau reducing learning rate to 3.8742047036066654e-05. 80/80
[=====] - 11s 143ms/step - loss: 0.0909 - iou_score: 0.8790 - val_loss: 0.1288 - val_iou_score: 0.8200 - lr: 4.3047e-05 Epoch 47/50 80/80 [=====] - ETA: 0s - loss: 0.0906 - iou_score: 0.8801
Epoch 47: saving model to model_2_e50/FPN_best_model_e47.h5 80/80 [=====] - 11s 140ms/step - loss: 0.0906 - iou_score: 0.8801 - val_loss: 0.1293 - val_iou_score: 0.8195 - lr: 3.8742e-05 Epoch 48/50 80/80
[=====] - ETA: 0s - loss: 0.0891 - iou_score: 0.8821 Epoch 48: saving model to model_2_e50/FPN_best_model_e48.h5 80/80 [=====] - 11s 140ms/step - loss: 0.0891 - iou_score: 0.8821 - val_loss: 0.1294 - val_iou_score: 0.8201 - lr: 3.8742e-05 Epoch 49/50 80/80 [=====] - ETA: 0s - loss: 0.0867 - iou_score: 0.8855 Epoch 49: saving model to model_2_e50/FPN_best_model_e49.h5 80/80
[=====] - 11s 140ms/step - loss: 0.0867 - iou_score: 0.8855 - val_loss: 0.1281 - val_iou_score: 0.8213 - lr: 3.8742e-05

Epoch 50/50 80/80 [=====] - ETA: 0s - loss: 0.0848 - iou_score: 0.8877 Epoch 50: saving model to model_2_e50/FPN_best_model_e50.h5 80/80 [=====] - 11s 139ms/step - loss: 0.0848 - iou_score: 0.8877 - val_loss: 0.1277 - val_iou_score: 0.8220 - lr: 3.8742e-05

Time Taken for training (sec): 589.2285118103027

In [48]:

```
FPN_model.load_weights('/content/r4_FPN_best_model_e50.h5')
```

In [49]:

```
# Run 1
optim = tf.keras.optimizers.Adam(3.8742e-05)

focal_loss = sm.losses.cce_dice_loss #cce_dice_loss = categorical_crossentropy + dice_loss

FPN_model.compile(optim, focal_loss, metrics=[iou_score])
```

In [50]:

```
datetime_stamp = datetime.now().strftime("%Y%m%d-%H%M%S")
logdir = os.path.join("logs", datetime_stamp)
# Create a timestamped log directory
```

```

print(aatetime_stamp)

# tensorboard = TensorBoard(log_dir=logdir)
tensorboard = TensorBoard(log_dir=logdir, histogram_freq=1, write_graph=True, write_grads=True)

checkpoint_m1 = ModelCheckpoint('model_2_e10/FPN_best_model_e50+{epoch:02d}.h5',
                               save_weights_only=True, save_best_only=False,
                               monitor='val_iou_score', verbose=1)

Reduce_LR_m1 = ReduceLROnPlateau(monitor='val_iou_score', factor=0.9, min_lr=0.00001, patience=5, verbose=1)

callbacks_m1 = [checkpoint_m1, Reduce_LR_m1, tensorboard]

start = time.time()
history_m1 = FPN_model.fit_generator(train_dataloader,
                                      steps_per_epoch=len(train_dataloader),
                                      epochs=50,
                                      validation_data=test_dataloader,
                                      callbacks=callbacks_m1)

stop = time.time()
print('Time Taken for training (sec): ', stop-start)

```

20220527-022928

WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
 WARNING:tensorflow:Model failed to serialize as JSON. Ignoring...
 Layer A2_Block has arguments ['self', 'f']
 in `__init__` and therefore must override `get_config()`.

Example:

```

class CustomLayer(keras.layers.Layer):
    def __init__(self, arg1, arg2):
        super().__init__()
        self.arg1 = arg1
        self.arg2 = arg2

    def get_config(self):
        config = super().get_config()
        config.update({
            "arg1": self.arg1,
            "arg2": self.arg2,
        })
        return config

```

Epoch 1/50

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:21: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

```

80/80 [=====] - ETA: 0s - loss: 0.0856 - iou_score: 0.8871
Epoch 1: saving model to model_2_e10/FPN_best_model_e50+01.h5
80/80 [=====] - 32s 161ms/step - loss: 0.0856 - iou_score: 0.8871 - val_loss: 0.1282 - val_iou_score: 0.8213 - lr: 3.8742e-05
Epoch 2/50
80/80 [=====] - ETA: 0s - loss: 0.0878 - iou_score: 0.8838
Epoch 2: saving model to model_2_e10/FPN_best_model_e50+02.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0878 - iou_score: 0.8838 - val_loss: 0.1357 - val_iou_score: 0.8116 - lr: 3.8742e-05
Epoch 3/50
80/80 [=====] - ETA: 0s - loss: 0.0882 - iou_score: 0.8827
Epoch 3: saving model to model_2_e10/FPN_best_model_e50+03.h5
80/80 [=====] - 13s 165ms/step - loss: 0.0882 - iou_score: 0.8827 - val_loss: 0.1257 - val_iou_score: 0.8248 - lr: 3.8742e-05
Epoch 4/50
80/80 [=====] - ETA: 0s - loss: 0.0847 - iou_score: 0.8880
Epoch 4: saving model to model_2_e10/FPN_best_model_e50+04.h5
80/80 [=====] - 11s 139ms/step - loss: 0.0847 - iou_score: 0.8880 - val_loss: 0.1252 - val_iou_score: 0.8256 - lr: 3.8742e-05
Epoch 5/50
80/80 [=====] - ETA: 0s - loss: 0.0836 - iou_score: 0.8900
Epoch 5: saving model to model_2_e10/FPN_best_model_e50+05.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0836 - iou_score: 0.8900 - val_loss: 0.1253 - val_iou_score: 0.8255 - lr: 3.8742e-05
Epoch 6/50
80/80 [=====] - ETA: 0s - loss: 0.0817 - iou score: 0.8925

```

```
Epoch 6: saving model to model_2_e10/FPN_best_model_e50+06.h5
80/80 [=====] - 11s 139ms/step - loss: 0.0817 - iou_score: 0.8925 - val_loss: 0.1240 - val_iou_score: 0.8272 - lr: 3.8742e-05
Epoch 7/50
80/80 [=====] - ETA: 0s - loss: 0.0811 - iou_score: 0.8931
Epoch 7: saving model to model_2_e10/FPN_best_model_e50+07.h5

Epoch 7: ReduceLROnPlateau reducing learning rate to 3.486780042294413e-05.
80/80 [=====] - 11s 140ms/step - loss: 0.0811 - iou_score: 0.8931 - val_loss: 0.1233 - val_iou_score: 0.8283 - lr: 3.8742e-05
Epoch 8/50
80/80 [=====] - ETA: 0s - loss: 0.0814 - iou_score: 0.8931
Epoch 8: saving model to model_2_e10/FPN_best_model_e50+08.h5
80/80 [=====] - 11s 138ms/step - loss: 0.0814 - iou_score: 0.8931 - val_loss: 0.1231 - val_iou_score: 0.8285 - lr: 3.4868e-05
Epoch 9/50
80/80 [=====] - ETA: 0s - loss: 0.0793 - iou_score: 0.8958
Epoch 9: saving model to model_2_e10/FPN_best_model_e50+09.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0793 - iou_score: 0.8958 - val_loss: 0.1220 - val_iou_score: 0.8302 - lr: 3.4868e-05
Epoch 10/50
80/80 [=====] - ETA: 0s - loss: 0.0770 - iou_score: 0.8992
Epoch 10: saving model to model_2_e10/FPN_best_model_e50+10.h5
80/80 [=====] - 12s 146ms/step - loss: 0.0770 - iou_score: 0.8992 - val_loss: 0.1220 - val_iou_score: 0.8302 - lr: 3.4868e-05
Epoch 11/50
80/80 [=====] - ETA: 0s - loss: 0.0767 - iou_score: 0.8995
Epoch 11: saving model to model_2_e10/FPN_best_model_e50+11.h5
80/80 [=====] - 11s 138ms/step - loss: 0.0767 - iou_score: 0.8995 - val_loss: 0.1226 - val_iou_score: 0.8295 - lr: 3.4868e-05
Epoch 12/50
80/80 [=====] - ETA: 0s - loss: 0.0768 - iou_score: 0.8997
Epoch 12: saving model to model_2_e10/FPN_best_model_e50+12.h5

Epoch 12: ReduceLROnPlateau reducing learning rate to 3.138102038064972e-05.
80/80 [=====] - 11s 140ms/step - loss: 0.0768 - iou_score: 0.8997 - val_loss: 0.1215 - val_iou_score: 0.8310 - lr: 3.4868e-05
Epoch 13/50
80/80 [=====] - ETA: 0s - loss: 0.0751 - iou_score: 0.9017
Epoch 13: saving model to model_2_e10/FPN_best_model_e50+13.h5
80/80 [=====] - 11s 144ms/step - loss: 0.0751 - iou_score: 0.9017 - val_loss: 0.1229 - val_iou_score: 0.8292 - lr: 3.1381e-05
Epoch 14/50
80/80 [=====] - ETA: 0s - loss: 0.0759 - iou_score: 0.9011
Epoch 14: saving model to model_2_e10/FPN_best_model_e50+14.h5
80/80 [=====] - 11s 139ms/step - loss: 0.0759 - iou_score: 0.9011 - val_loss: 0.1229 - val_iou_score: 0.8288 - lr: 3.1381e-05
Epoch 15/50
80/80 [=====] - ETA: 0s - loss: 0.0749 - iou_score: 0.9027
Epoch 15: saving model to model_2_e10/FPN_best_model_e50+15.h5
80/80 [=====] - 11s 139ms/step - loss: 0.0749 - iou_score: 0.9027 - val_loss: 0.1212 - val_iou_score: 0.8317 - lr: 3.1381e-05
Epoch 16/50
80/80 [=====] - ETA: 0s - loss: 0.0736 - iou_score: 0.9045
Epoch 16: saving model to model_2_e10/FPN_best_model_e50+16.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0736 - iou_score: 0.9045 - val_loss: 0.1217 - val_iou_score: 0.8310 - lr: 3.1381e-05
Epoch 17/50
80/80 [=====] - ETA: 0s - loss: 0.0726 - iou_score: 0.9057
Epoch 17: saving model to model_2_e10/FPN_best_model_e50+17.h5

Epoch 17: ReduceLROnPlateau reducing learning rate to 2.824291768774856e-05.
80/80 [=====] - 11s 141ms/step - loss: 0.0726 - iou_score: 0.9057 - val_loss: 0.1210 - val_iou_score: 0.8323 - lr: 3.1381e-05
Epoch 18/50
80/80 [=====] - ETA: 0s - loss: 0.0732 - iou_score: 0.9050
Epoch 18: saving model to model_2_e10/FPN_best_model_e50+18.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0732 - iou_score: 0.9050 - val_loss: 0.1213 - val_iou_score: 0.8314 - lr: 2.8243e-05
Epoch 19/50
80/80 [=====] - ETA: 0s - loss: 0.0720 - iou_score: 0.9067
Epoch 19: saving model to model_2_e10/FPN_best_model_e50+19.h5
80/80 [=====] - 11s 142ms/step - loss: 0.0720 - iou_score: 0.9067 - val_loss: 0.1208 - val_iou_score: 0.8324 - lr: 2.8243e-05
Epoch 20/50
80/80 [=====] - ETA: 0s - loss: 0.0719 - iou_score: 0.9070
Epoch 20: saving model to model_2_e10/FPN_best_model_e50+20.h5
```

```
Epoch 20: saving model to model_2_e10/FPN_best_model_e50+20.h5
80/80 [=====] - 11s 139ms/step - loss: 0.0719 - iou_score: 0.9070 - val_loss: 0.1219 - val_iou_score: 0.8306 - lr: 2.8243e-05
Epoch 21/50
80/80 [=====] - ETA: 0s - loss: 0.0708 - iou_score: 0.9088
Epoch 21: saving model to model_2_e10/FPN_best_model_e50+21.h5
80/80 [=====] - 12s 147ms/step - loss: 0.0708 - iou_score: 0.9088 - val_loss: 0.1187 - val_iou_score: 0.8352 - lr: 2.8243e-05
Epoch 22/50
80/80 [=====] - ETA: 0s - loss: 0.0695 - iou_score: 0.9106
Epoch 22: saving model to model_2_e10/FPN_best_model_e50+22.h5

Epoch 22: ReduceLROnPlateau reducing learning rate to 2.5418625591555612e-05.
80/80 [=====] - 11s 141ms/step - loss: 0.0695 - iou_score: 0.9106 - val_loss: 0.1207 - val_iou_score: 0.8326 - lr: 2.8243e-05
Epoch 23/50
80/80 [=====] - ETA: 0s - loss: 0.0685 - iou_score: 0.9120
Epoch 23: saving model to model_2_e10/FPN_best_model_e50+23.h5
80/80 [=====] - 12s 149ms/step - loss: 0.0685 - iou_score: 0.9120 - val_loss: 0.1201 - val_iou_score: 0.8340 - lr: 2.5419e-05
Epoch 24/50
80/80 [=====] - ETA: 0s - loss: 0.0684 - iou_score: 0.9123
Epoch 24: saving model to model_2_e10/FPN_best_model_e50+24.h5
80/80 [=====] - 11s 142ms/step - loss: 0.0684 - iou_score: 0.9123 - val_loss: 0.1218 - val_iou_score: 0.8311 - lr: 2.5419e-05
Epoch 25/50
80/80 [=====] - ETA: 0s - loss: 0.0684 - iou_score: 0.9123
Epoch 25: saving model to model_2_e10/FPN_best_model_e50+25.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0684 - iou_score: 0.9123 - val_loss: 0.1204 - val_iou_score: 0.8339 - lr: 2.5419e-05
Epoch 26/50
80/80 [=====] - ETA: 0s - loss: 0.0675 - iou_score: 0.9136
Epoch 26: saving model to model_2_e10/FPN_best_model_e50+26.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0675 - iou_score: 0.9136 - val_loss: 0.1201 - val_iou_score: 0.8342 - lr: 2.5419e-05
Epoch 27/50
80/80 [=====] - ETA: 0s - loss: 0.0671 - iou_score: 0.9142
Epoch 27: saving model to model_2_e10/FPN_best_model_e50+27.h5

Epoch 27: ReduceLROnPlateau reducing learning rate to 2.2876762704981958e-05.
80/80 [=====] - 11s 140ms/step - loss: 0.0671 - iou_score: 0.9142 - val_loss: 0.1215 - val_iou_score: 0.8328 - lr: 2.5419e-05
Epoch 28/50
80/80 [=====] - ETA: 0s - loss: 0.0681 - iou_score: 0.9126
Epoch 28: saving model to model_2_e10/FPN_best_model_e50+28.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0681 - iou_score: 0.9126 - val_loss: 0.1191 - val_iou_score: 0.8355 - lr: 2.2877e-05
Epoch 29/50
80/80 [=====] - ETA: 0s - loss: 0.0670 - iou_score: 0.9144
Epoch 29: saving model to model_2_e10/FPN_best_model_e50+29.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0670 - iou_score: 0.9144 - val_loss: 0.1214 - val_iou_score: 0.8329 - lr: 2.2877e-05
Epoch 30/50
80/80 [=====] - ETA: 0s - loss: 0.0665 - iou_score: 0.9144
Epoch 30: saving model to model_2_e10/FPN_best_model_e50+30.h5
80/80 [=====] - 11s 142ms/step - loss: 0.0665 - iou_score: 0.9144 - val_loss: 0.1216 - val_iou_score: 0.8334 - lr: 2.2877e-05
Epoch 31/50
80/80 [=====] - ETA: 0s - loss: 0.0648 - iou_score: 0.9164
Epoch 31: saving model to model_2_e10/FPN_best_model_e50+31.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0648 - iou_score: 0.9164 - val_loss: 0.1245 - val_iou_score: 0.8306 - lr: 2.2877e-05
Epoch 32/50
80/80 [=====] - ETA: 0s - loss: 0.0635 - iou_score: 0.9178
Epoch 32: saving model to model_2_e10/FPN_best_model_e50+32.h5

Epoch 32: ReduceLROnPlateau reducing learning rate to 2.0589085943356624e-05.
80/80 [=====] - 11s 142ms/step - loss: 0.0635 - iou_score: 0.9178 - val_loss: 0.1231 - val_iou_score: 0.8319 - lr: 2.2877e-05
Epoch 33/50
80/80 [=====] - ETA: 0s - loss: 0.0622 - iou_score: 0.9187
Epoch 33: saving model to model_2_e10/FPN_best_model_e50+33.h5
80/80 [=====] - 11s 142ms/step - loss: 0.0622 - iou_score: 0.9187 - val_loss: 0.1243 - val_iou_score: 0.8311 - lr: 2.0589e-05
Epoch 34/50
80/80 [=====] - ETA: 0s - loss: 0.0604 - iou_score: 0.9211
Epoch 34: saving model to model_2_e10/FPN_best_model_e50+34.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0604 - iou_score: 0.9211 - val_loss:
```

```
00/00 [=====] - 11s 141ms/step - loss: 0.0584 - iou_score: 0.9211 - val_loss: 0.1224 - val_iou_score: 0.8330 - lr: 2.0589e-05
Epoch 35/50
80/80 [=====] - ETA: 0s - loss: 0.0586 - iou_score: 0.9235
Epoch 35: saving model to model_2_e10/FPN_best_model_e50+35.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0586 - iou_score: 0.9235 - val_loss: 0.1242 - val_iou_score: 0.8304 - lr: 2.0589e-05
Epoch 36/50
80/80 [=====] - ETA: 0s - loss: 0.0576 - iou_score: 0.9247
Epoch 36: saving model to model_2_e10/FPN_best_model_e50+36.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0576 - iou_score: 0.9247 - val_loss: 0.1221 - val_iou_score: 0.8337 - lr: 2.0589e-05
Epoch 37/50
80/80 [=====] - ETA: 0s - loss: 0.0560 - iou_score: 0.9270
Epoch 37: saving model to model_2_e10/FPN_best_model_e50+37.h5

Epoch 37: ReduceLROnPlateau reducing learning rate to 1.85301778401481e-05.
80/80 [=====] - 12s 144ms/step - loss: 0.0560 - iou_score: 0.9270 - val_loss: 0.1284 - val_iou_score: 0.8258 - lr: 2.0589e-05
Epoch 38/50
80/80 [=====] - ETA: 0s - loss: 0.0674 - iou_score: 0.9097
Epoch 38: saving model to model_2_e10/FPN_best_model_e50+38.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0674 - iou_score: 0.9097 - val_loss: 0.1231 - val_iou_score: 0.8318 - lr: 1.8530e-05
Epoch 39/50
80/80 [=====] - ETA: 0s - loss: 0.0612 - iou_score: 0.9190
Epoch 39: saving model to model_2_e10/FPN_best_model_e50+39.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0612 - iou_score: 0.9190 - val_loss: 0.1298 - val_iou_score: 0.8248 - lr: 1.8530e-05
Epoch 40/50
80/80 [=====] - ETA: 0s - loss: 0.0612 - iou_score: 0.9188
Epoch 40: saving model to model_2_e10/FPN_best_model_e50+40.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0612 - iou_score: 0.9188 - val_loss: 0.1244 - val_iou_score: 0.8320 - lr: 1.8530e-05
Epoch 41/50
80/80 [=====] - ETA: 0s - loss: 0.0562 - iou_score: 0.9260
Epoch 41: saving model to model_2_e10/FPN_best_model_e50+41.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0562 - iou_score: 0.9260 - val_loss: 0.1225 - val_iou_score: 0.8342 - lr: 1.8530e-05
Epoch 42/50
80/80 [=====] - ETA: 0s - loss: 0.0527 - iou_score: 0.9301
Epoch 42: saving model to model_2_e10/FPN_best_model_e50+42.h5

Epoch 42: ReduceLROnPlateau reducing learning rate to 1.667716005613329e-05.
80/80 [=====] - 11s 142ms/step - loss: 0.0527 - iou_score: 0.9301 - val_loss: 0.1253 - val_iou_score: 0.8305 - lr: 1.8530e-05
Epoch 43/50
80/80 [=====] - ETA: 0s - loss: 0.0492 - iou_score: 0.9332
Epoch 43: saving model to model_2_e10/FPN_best_model_e50+43.h5
80/80 [=====] - 11s 143ms/step - loss: 0.0492 - iou_score: 0.9332 - val_loss: 0.1231 - val_iou_score: 0.8332 - lr: 1.6677e-05
Epoch 44/50
80/80 [=====] - ETA: 0s - loss: 0.0449 - iou_score: 0.9382
Epoch 44: saving model to model_2_e10/FPN_best_model_e50+44.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0449 - iou_score: 0.9382 - val_loss: 0.1220 - val_iou_score: 0.8346 - lr: 1.6677e-05
Epoch 45/50
80/80 [=====] - ETA: 0s - loss: 0.0436 - iou_score: 0.9398
Epoch 45: saving model to model_2_e10/FPN_best_model_e50+45.h5
80/80 [=====] - 11s 139ms/step - loss: 0.0436 - iou_score: 0.9398 - val_loss: 0.1228 - val_iou_score: 0.8333 - lr: 1.6677e-05
Epoch 46/50
80/80 [=====] - ETA: 0s - loss: 0.0431 - iou_score: 0.9406
Epoch 46: saving model to model_2_e10/FPN_best_model_e50+46.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0431 - iou_score: 0.9406 - val_loss: 0.1220 - val_iou_score: 0.8346 - lr: 1.6677e-05
Epoch 47/50
80/80 [=====] - ETA: 0s - loss: 0.0424 - iou_score: 0.9415
Epoch 47: saving model to model_2_e10/FPN_best_model_e50+47.h5

Epoch 47: ReduceLROnPlateau reducing learning rate to 1.50094445416471e-05.
80/80 [=====] - 11s 141ms/step - loss: 0.0424 - iou_score: 0.9415 - val_loss: 0.1213 - val_iou_score: 0.8357 - lr: 1.6677e-05
Epoch 48/50
80/80 [=====] - ETA: 0s - loss: 0.0423 - iou_score: 0.9418
Epoch 48: saving model to model_2_e10/FPN_best_model_e50+48.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0423 - iou_score: 0.9418 - val_loss: 0.1225 - val_iou_score: 0.8342 - lr: 1.5009e-05
```

```

0.1220 - val_iou_score: 0.0542 - lr: 1.5009e-05
Epoch 49/50
80/80 [=====] - ETA: 0s - loss: 0.0415 - iou_score: 0.9428
Epoch 49: saving model to model_2_e10/FPN_best_model_e50+49.h5
80/80 [=====] - 11s 140ms/step - loss: 0.0415 - iou_score: 0.9428 - val_loss:
0.1220 - val_iou_score: 0.8346 - lr: 1.5009e-05
Epoch 50/50
80/80 [=====] - ETA: 0s - loss: 0.0406 - iou_score: 0.9439
Epoch 50: saving model to model_2_e10/FPN_best_model_e50+50.h5
80/80 [=====] - 11s 141ms/step - loss: 0.0406 - iou_score: 0.9439 - val_loss:
0.1219 - val_iou_score: 0.8350 - lr: 1.5009e-05
Time Taken for training (sec): 614.61580991745

```

In [51]:

```
# # http://localhost:6006/
%load_ext tensorboard
%tensorboard --logdir logs --host localhost
```

In [52]:

```
# index of best validation score
np.argmax(history_m1.history['val_iou_score'])
```

Out[52]:

46

Predicting patches using Best A2FPN weights

In [51]:

```
!wget --header="Host: doc-0c-1o-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Cookie: ATUH_3hse9g93rcadk3jhj6f018c5j3o1r79s_nonce=bbi3ik3cfsc4" --header="Connection: keep-alive" "https://doc-0c-1o-docs.googleusercontent.com/docs/securesc/brr5e7j2b01vas75j78kqje1o8uv/gmqabt4q872knjon117e96s7s0mshqss/1653618075000/14063734026192868486/14063734026192868486/159o1vgXb0LdRc9TfEm2TKbkVkb7vJESb?e=download&ax=ACxEAsZgU5LNvnyaU3Ig-Zhx_VLZEOCxsy1-pm9QYZow5YfkkbXI-zsfNhWptmEjJ7o_QyFiR1UzsTrXYpLw79GedVGL37_f1XUT-zhGyh4SjoAu7Y6jNgleLoB6RjMNIW-AA89U9q9Hrh-AET7qzIBDI6WqO-zNvHqgskQpqjNru7o-DMfUJdoKmT75YFBzOY-WbrWayEaEqfsqpf1yznHpUKQI7cKHchhRS2Y6kvY8TBWWz21_PhWddokDop1jsE-PmDLPeikyCgDR17VoUvwU4xc8I9LuxXkXpyIecEyd0o2BsctJVXlpSJWQGar60TkDNbgX Achw691TWIxPrLrIAt729Z6eW9inbPBejLmQeLXva9CwK-6PE8NoVPjc4_YwTkR08oyj_OZroZ6Xy9eF-X3mgvvuoEn41WOkEuT1TaP6exw3BF1kZ7VsTm4byHN1uLZEz0WEI73t36RgJvxoAa_x3bBmuy_Et8t1SOk3KhOHdm9lXv121bRfPufBMry0NmmxAU7P_M1iCaGfmFU28FaBXyvZL1w_vqoVkyLkPKiB0Ye7BbgYzxlpQIPSSjA-KXvqMsXVrte9e-nR3gCpg9G2m_XyJVUKNK8S1yNHVUVDDqPt8-evKHxeNpluo2BKC3i4rh4mN5j6eiflduYACYRiCT_1PYXEisUsXRdpYce-W3-wa8C7zBseFsXEGu8jXgkaJ5lw&authuser=0&nonce=bbi3ik3cfsc4&user=14063734026192868486&hash=84g4viu3fpt0apggngqbd6k1vitfqoquv" -c -O 'r4_FPN_best_model_e50.h5'
```

```
--2022-05-27 02:22:07-- https://doc-0c-1o-docs.googleusercontent.com/docs/securesc/brr5e7j2b01vas75j784c78kqje1o8uv/gmqabt4q872knjon117e96s7s0mshqss/1653618075000/14063734026192868486/14063734026192868486/159o1vgXb0LdRc9TfEm2TKbkVkb7vJESb?e=download&ax=ACxEAsZgU5LNvnyaU3Ig-Zhx_VLZEOCxsy1-pm9QYZow5YfkkbXI-zsfNhWptmEjJ7o_QyFiR1UzsTrXYpLw79GedVGL37_f1XUT-zhGyh4SjoAu7Y6jNgleLoB6RjMNIW-AA89U9q9Hrh-AET7qzIBDI6WqO-zNvHqgskQpqjNru7o-DMfUJdoKmT75YFBzOY-WbrWayEaEqfsqpf1yznHpUKQI7cKHchhRS2Y6kvY8TBWWz21_PhWddokDop1jsE-PmDLPeikyCgDR17VoUvwU4xc8I9LuxXkXpyIecEyd0o2BsctJVXlpSJWQGar60TkDNbgX Achw691TWIxPrLrIAt729Z6eW9inbPBejLmQeLXva9CwK-6PE8NoVPjc4_YwTkR08oyj_OZroZ6Xy9eF-X3mgvvuoEn41WOkEuT1TaP6exw3BF1kZ7VsTm4byHN1uLZEz0WEI73t36RgJvxoAa_x3bBmuy_Et8t1SOk3KhOHdm9lXv121bRfPufBMry0NmmxAU7P_M1iCaGfmFU28FaBXyvZL1w_vqoVkyLkPKiB0Ye7BbgYzxlpQIPSSjA-KXvqMsXVrte9e-nR3gCpg9G2m_XyJVUKNK8S1yNHVUVDDqPt8-evKHxeNpluo2BKC3i4rh4mN5j6eiflduYACYRiCT_1PYXEisUsXRdpYce-W3-wa8C7zBseFsXEGu8jXgkaJ5lw&authuser=0&nonce=bbi3ik3cfsc4&user=14063734026192868486&hash=84g4viu3fpt0apggngqbd6k1vitfqouqv
Resolving doc-0c-1o-docs.googleusercontent.com (doc-0c-1o-docs.googleusercontent.com)... 108.177.98.132, 2607:f8b0:400e:c06::84
Connecting to doc-0c-1o-docs.googleusercontent.com (doc-0c-1o-docs.googleusercontent.com) |108.177.98.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89482192 (85M) [application/octet-stream]
Saving to: 'r4_FPN_best_model_e50.h5'

r4_FPN_best_model_e 100%[=====] 85.34M 73.2MB/s in 1.2s
```

```
2022-05-21 02:22:09 (13.2 MB/s) - r4_FPN_best_model_e50.h5 saved [89482192/89482192]
```

In [52]:

```
# Loading saved model weights
FPN_model.load_weights('/content/r4_FPN_best_model_e50.h5')
```

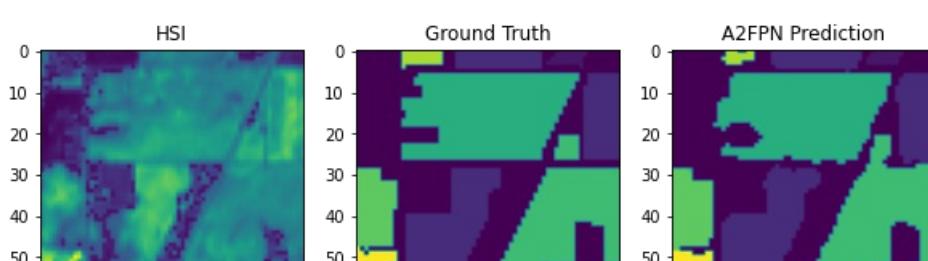
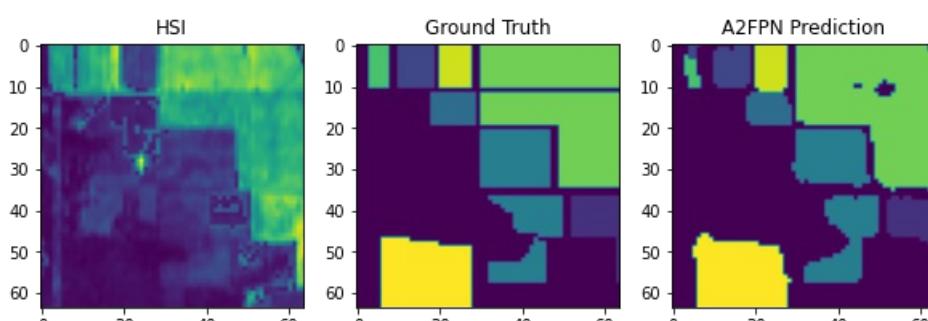
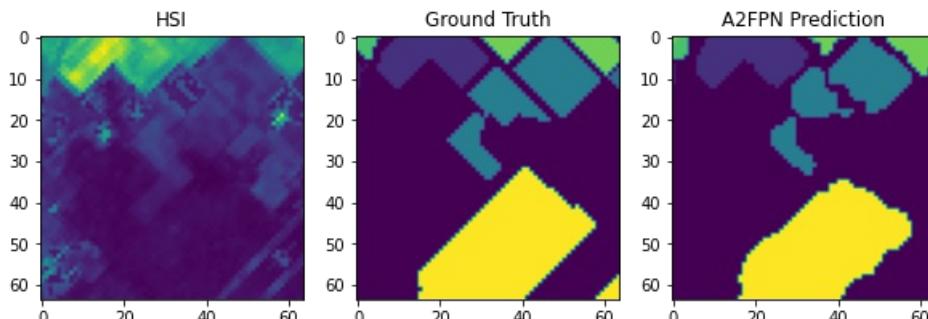
In [53]:

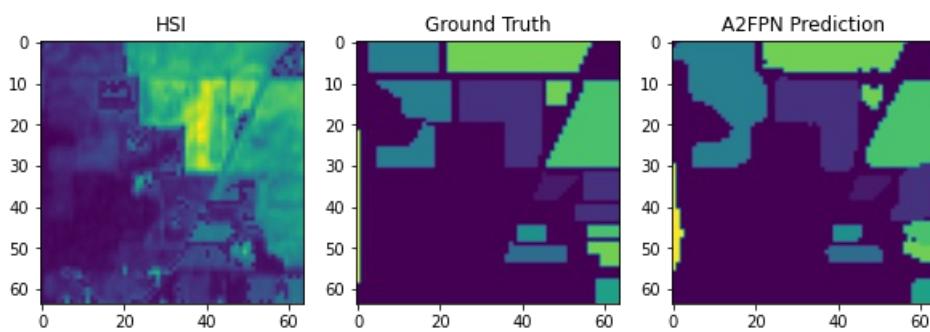
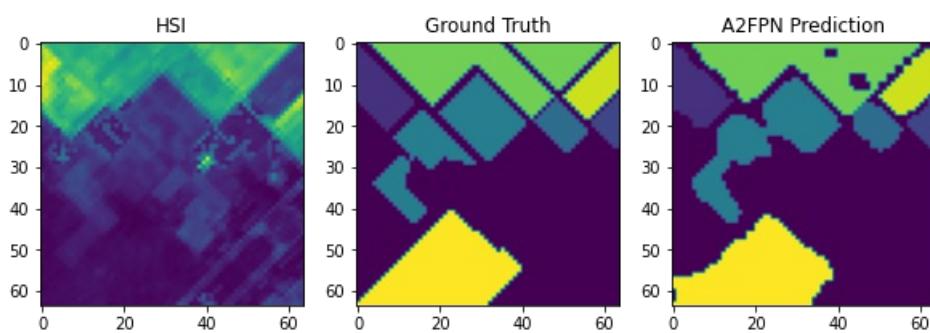
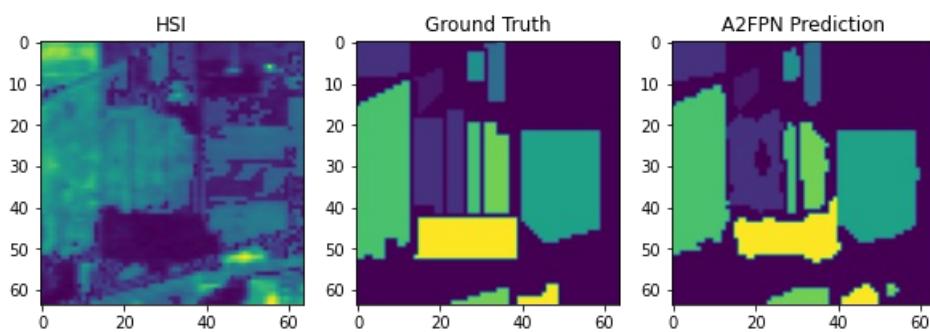
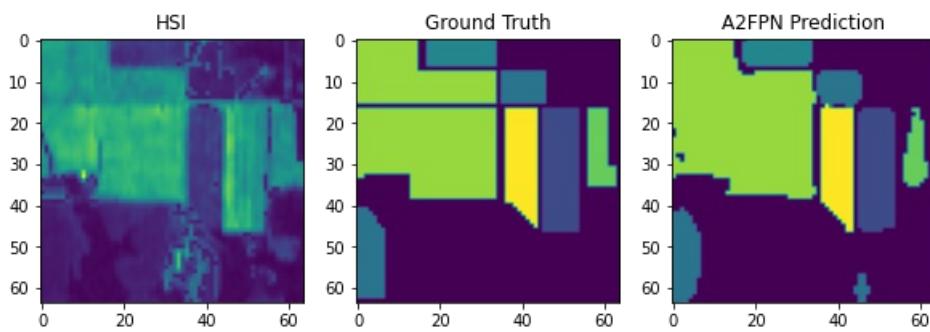
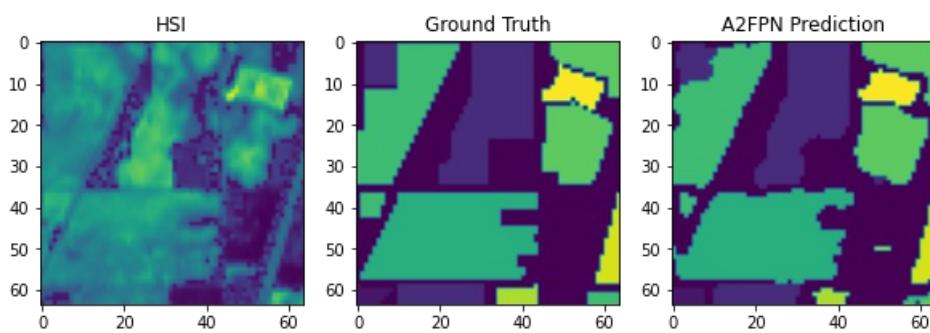
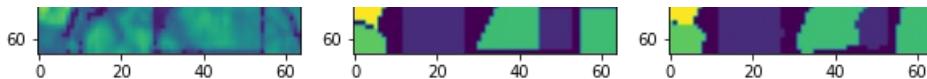
```
# Plotting Model prediction of segmentation alongside HSI and Ground Truth
i=0
for im, gt in zip(X_test[20:100],y_test[20:100]):

    # model prediction
    pred = FPN_model.predict(im[np.newaxis,:,:,:])

    # generating the image based on the max probability of particular class
    prediction = np.argmax(pred, axis=-1)

    # plotting HSI image vs ground truth vs prediction
    plt.figure(figsize=(10,6))
    plt.subplot(131)
    plt.imshow(im[:, :, 20])
    plt.title('HSI')
    plt.subplot(132)
    plt.imshow(gt)
    plt.title('Ground Truth')
    plt.subplot(133)
    plt.imshow(prediction[0])
    plt.title('A2FPN Prediction')
    plt.show()
    i+=1
if(i>10):
    break
```

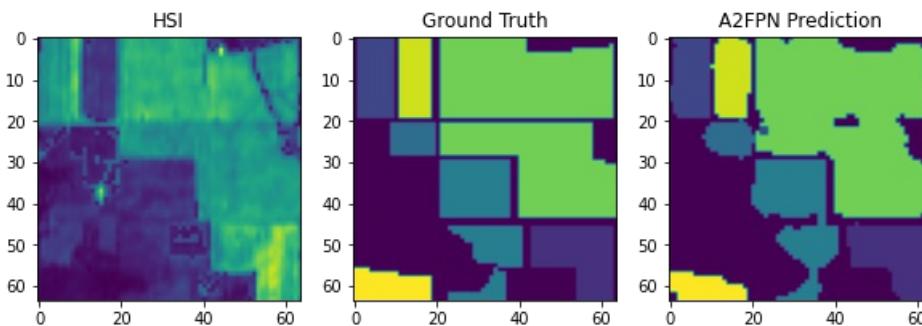
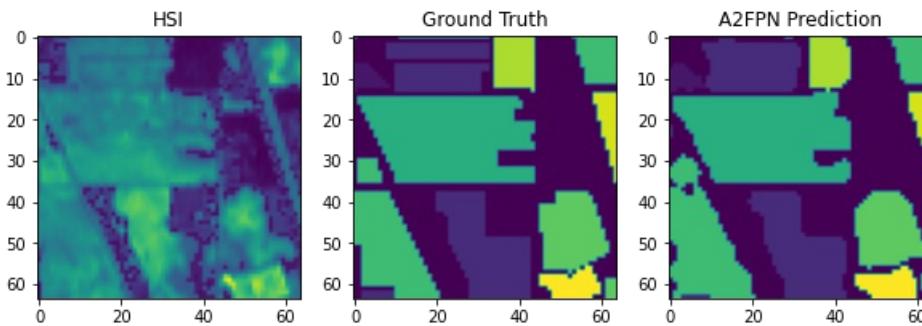
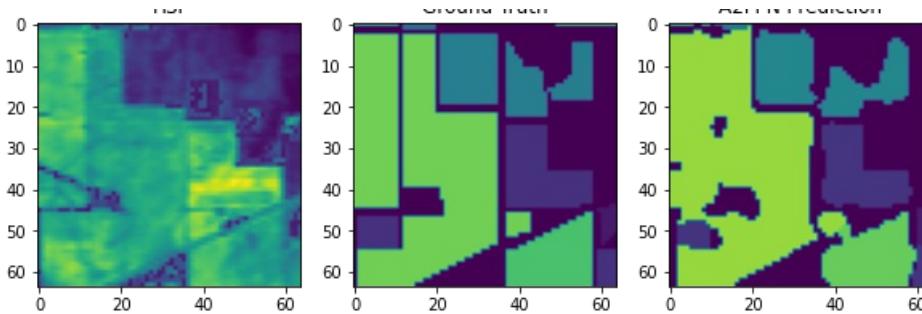




HSI

Ground Truth

A2FPN Prediction



A2FPN prediction for complete image

Generating the segmentation of original image (145x145) from patches

In [54]:

```
HSI_orig_patch = img_patch_list_new[0]
HSI_orig_patch.shape
```

Out[54]:

(10, 10, 64, 64, 95)

In [55]:

```
# Loading data associated with the original image (145x145)
HSI_orig_dataset = []
for i in range(HSI_orig_patch.shape[0]):
    for j in range(HSI_orig_patch.shape[1]):
        single_patch = HSI_orig_patch[i][j]
        single_patch = Std_scaler.transform(single_patch.reshape(-1, single_patch.shape[-1])).reshape(single_patch.shape)
        HSI_orig_dataset.append(single_patch)
```

In [56]:

```
# Converting original patch list to numpy array
HSI_orig_dataset = np.array(HSI_orig_dataset)
```

In [57]:

```
HSI_orig_dataset.shape
```

Out[57]:

```
(100, 64, 64, 95)
```

In [58]:

```
# predicting for individual patch
pred = FPN_model.predict(HSI_orig_dataset)
prediction = np.argmax(pred, axis=-1)
```

In [59]:

```
pred.shape
```

Out[59]:

```
(100, 64, 64, 17)
```

In [60]:

```
# individual patch is combined to form a grid of patches
grid = 0
img_pred = np.zeros((10, 10, 64, 64))
for i in range(10):
    for j in range(10):
        img_pred[i][j] = prediction[grid]
        grid+=1
```

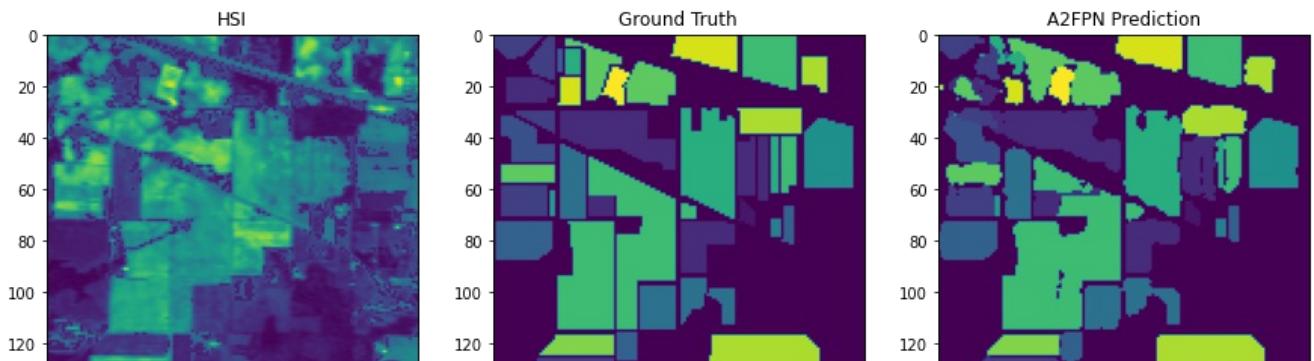
Unpatchified prediction

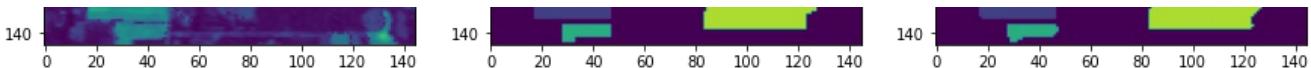
In [61]:

```
# converting the predicted patches into complete image using unpatchify
HSI_orig_pred = patch.unpatchify(img_pred, (145,145))
```

In [62]:

```
# plotting comparison of HSI vs Ground truth vs unet_m1 predictions
plt.figure(figsize=(15,15))
plt.subplot(131)
plt.imshow(img[:, :, 30])
plt.title('HSI')
plt.subplot(132)
plt.imshow(img_gt)
plt.title('Ground Truth')
plt.subplot(133)
plt.imshow(HSI_orig_pred)
plt.title('A2FPN Prediction')
plt.show()
```





Note: In unpatchify method, each patch at the overlapping regions are replaced by next patch. Alternative approach for stitching all patches is presented below.

Prediction based on max score of patches

Here the segmentation is generated by constructing the matrix of size (145, 145, 100*17) where model prediction probabilities(64x64x17) of each patch are placed along third axis in a manner mentioned below:

- First patch(predictions) will be placed at (0,0,0)
- Second patch(predictions) will be placed at (0,9,17)
- Third patch(predictions) will be placed at (0,18,34) -...
- Last patch(predictions) will be placed at (137,137,1684)

This is done to consider max probability from multiple prediction for the overlapping regions. In this way the best class is selected at overlapping regions by using argmax along third axis and modulo operator for 17

In [63]:

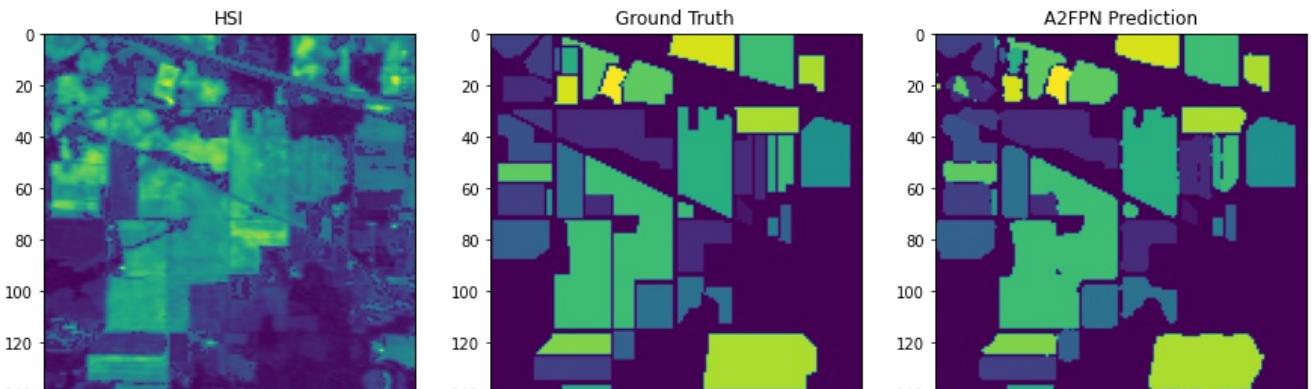
```
# Generating the 3D probabilities grid of all patches associated with full image.
grid = 0
grp = 0
img_prediction = np.zeros((145, 145, 100*17))
for i in range(10):
    for j in range(10):
        img_prediction[i*9:i*9+64,
                      j*9:j*9+64,
                      grp:grp+17] = pred[grid]
    grid+=1
    grp+=17
```

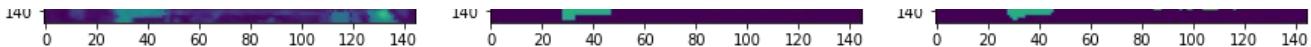
In [64]:

```
# Identifying the classes of each pixel from probabilities values of all patches corresponding to image (145x145)
prediction = np.argmax(img_prediction, axis=-1)%17
```

In [65]:

```
# Plotting the segmentation after identifying the best class for overlapping patches
plt.figure(figsize=(15,15))
plt.subplot(131)
plt.imshow(img[:, :, 30])
plt.title('HSI')
plt.subplot(132)
plt.imshow(img_gt)
plt.title('Ground Truth')
plt.subplot(133)
plt.imshow(prediction)
plt.title('A2FPN Prediction')
plt.show()
```





We can observe that the segmentation is better than the unpatchify generated image.

Full image prediction score (F1 and kappa)

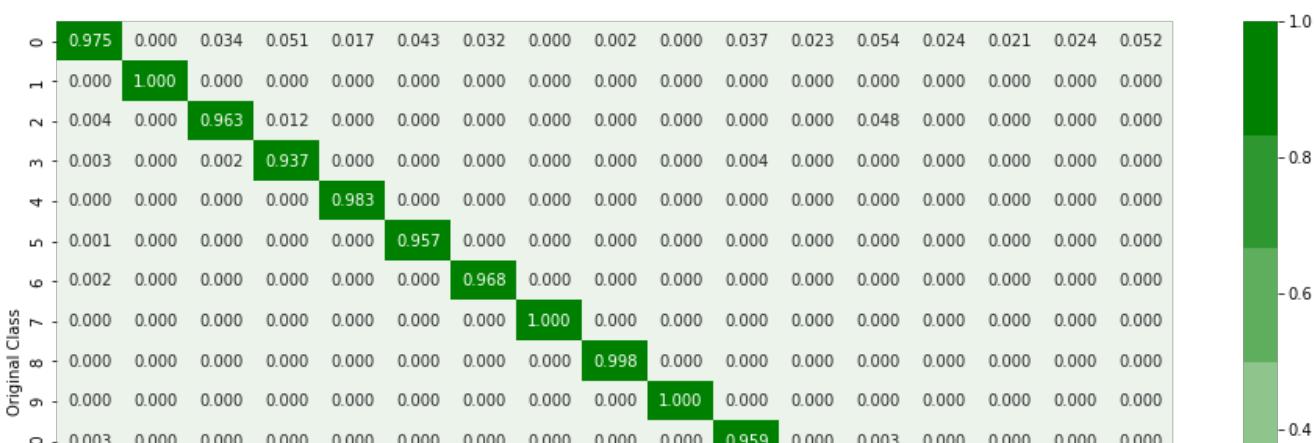
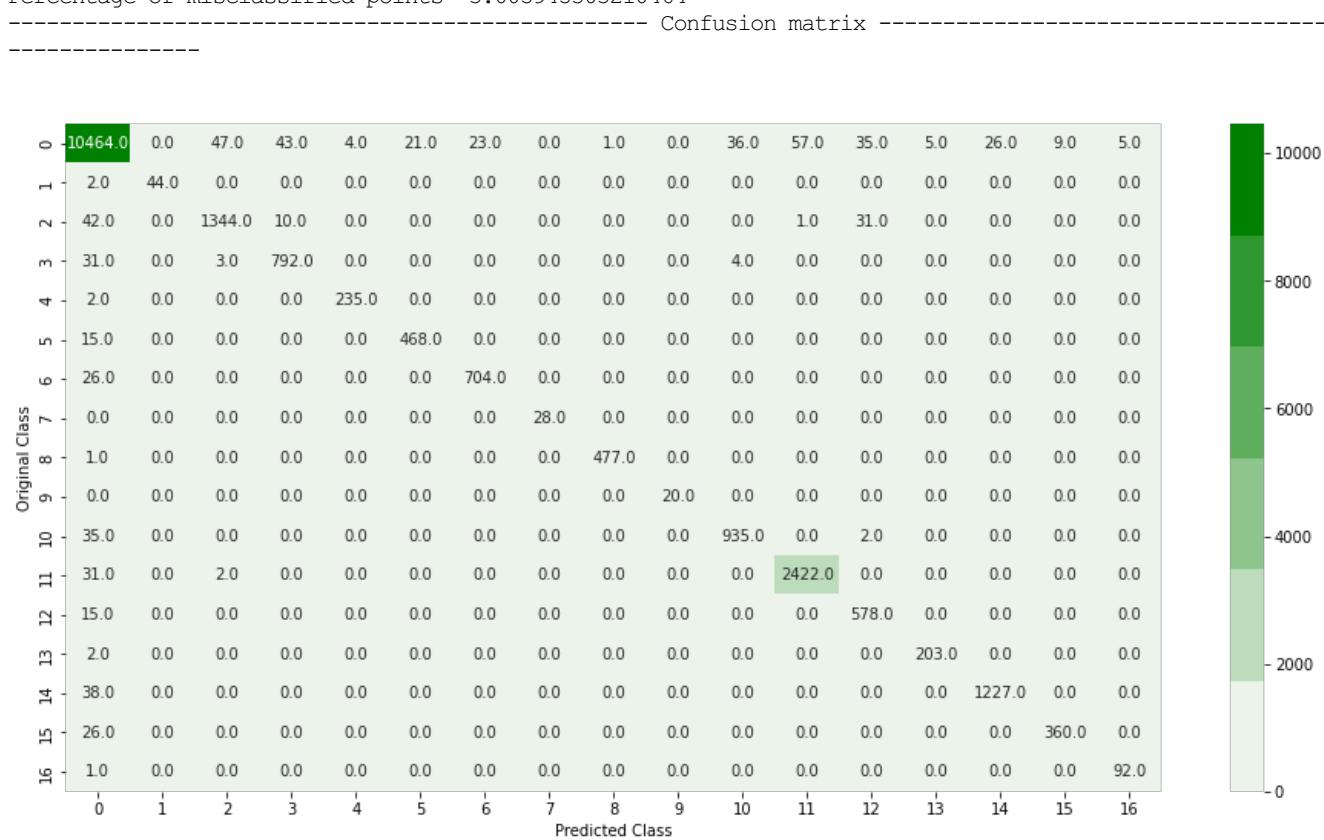
In [66]:

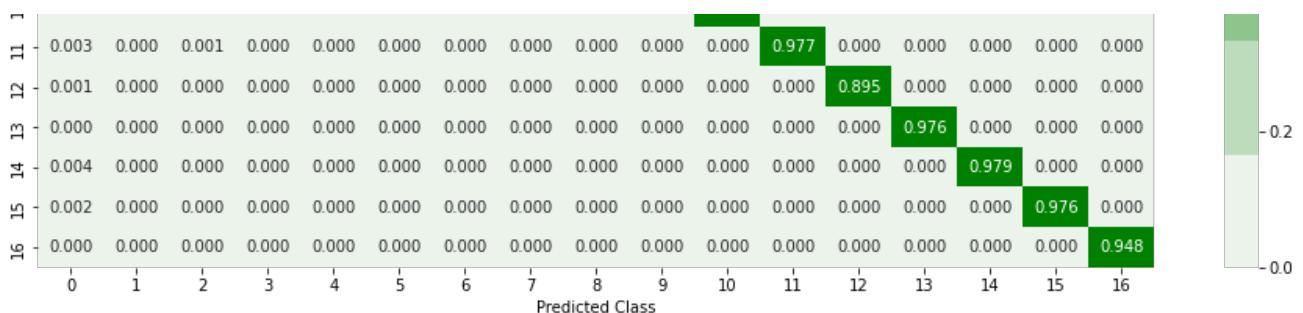
```
# Flattening the ground truths and predictions (145x145 image) for score evaluation
y = img_gt.flatten()
y_hat = prediction.flatten()
```

In [67]:

```
plot_confusion_matrix_2(y,y_hat)
```

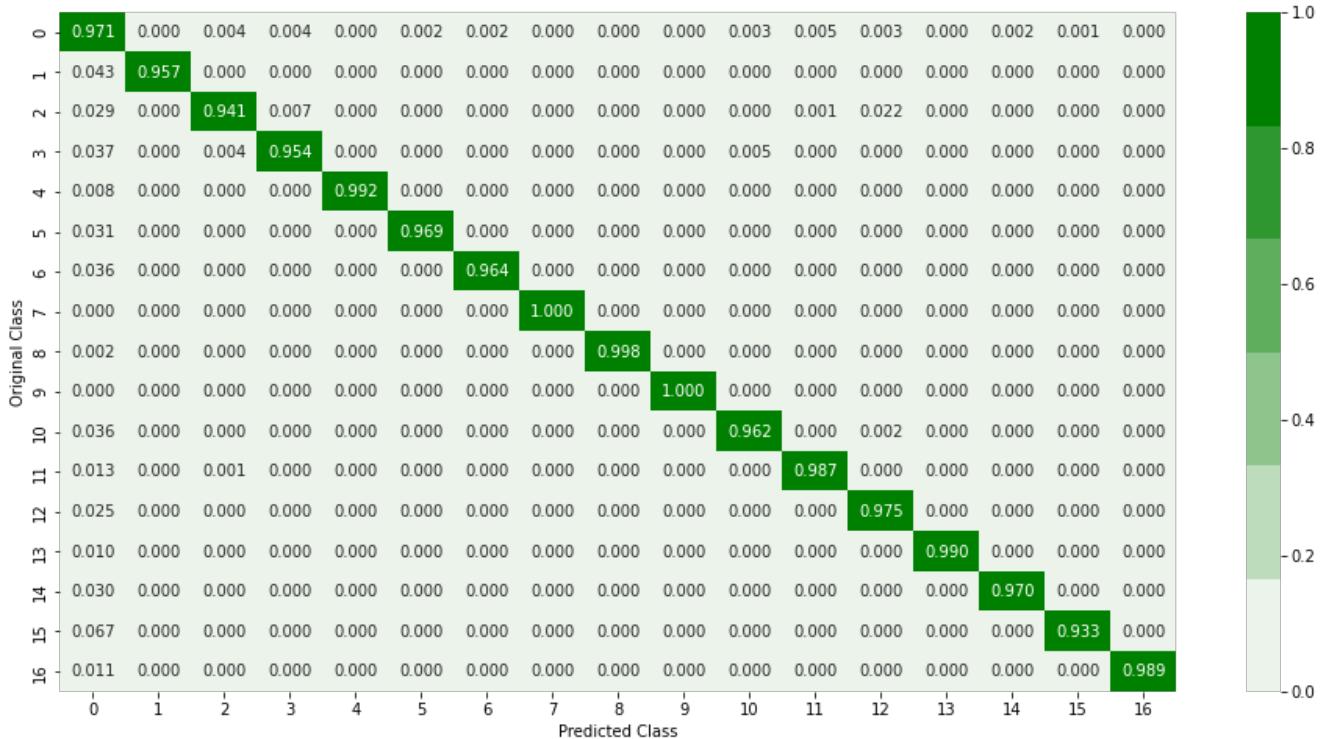
Confusion / Precision / Recall matrix
Percentage of misclassified points 3.005945303210464





Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

----- Recall matrix -----



Sum of rows in recall matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

micro F1 score : 0.9699405469678953

Average Accuracy : 0.9735911710136992

In [68]:

```
F1_A2FPN = f1_score(y,y_hat,average='micro')
print('micro F1 score of pretrained A2FPN model for full image : ',F1_A2FPN)
kappa_A2FPN = cohen_kappa_score(y,y_hat)
print('kappa score of pretrained A2FPN model for full image : ',kappa_A2FPN)
```

micro F1 score of pretrained A2FPN model for full image : 0.9699405469678953
kappa score of pretrained A2FPN model for full image : 0.957609937934343

Validation set score

Score evaluation for the test split to understand the performance of predicting the patches

In [69]:

```
X_test.shape,y_test.shape
```

Out[69]:

```
((200, 64, 64, 95), (200, 64, 64))
```

In [70]:

```
pred_test = FPN_model.predict(X_test)
prediction_test = np.argmax(pred_test, axis=-1)
```

In [71]:

```
prediction_test.shape
```

Out[71]:

```
(200, 64, 64)
```

In [72]:

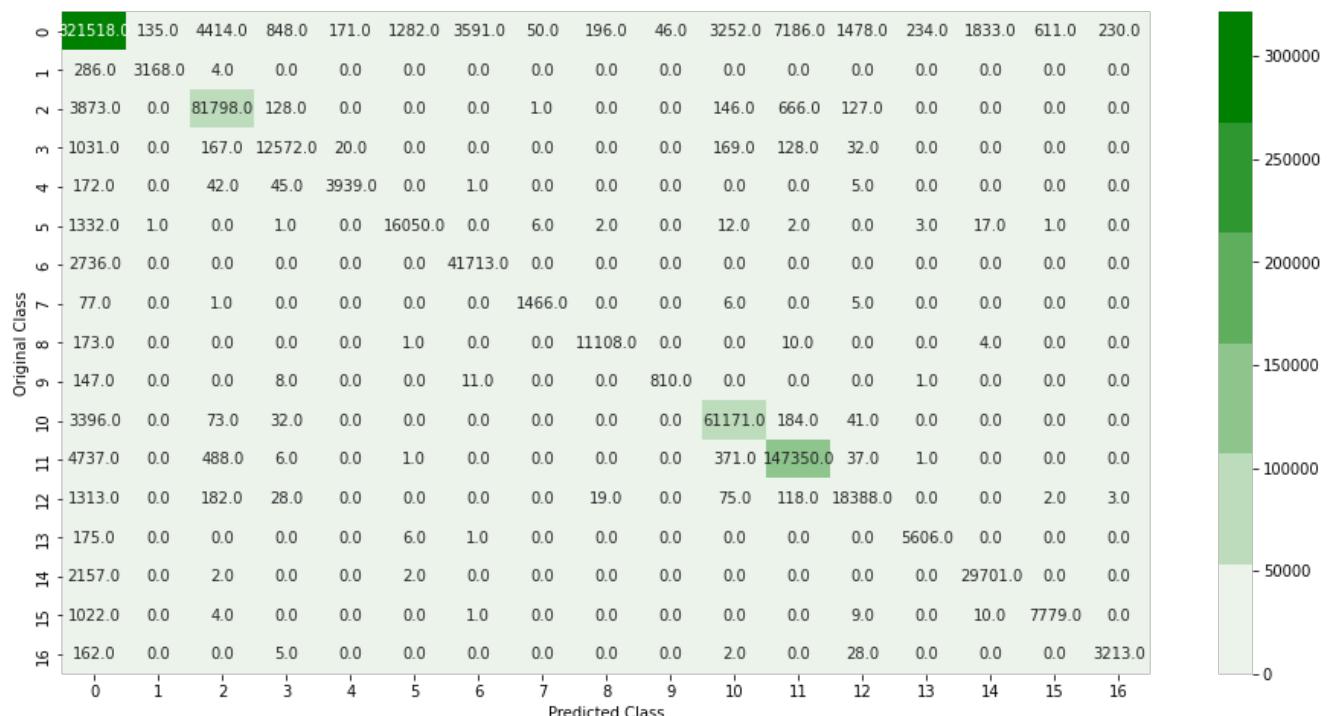
```
# Flattening the prediction of validation/test set
y_val = y_test.flatten()
y_hat_val = prediction_test.flatten()
```

In [73]:

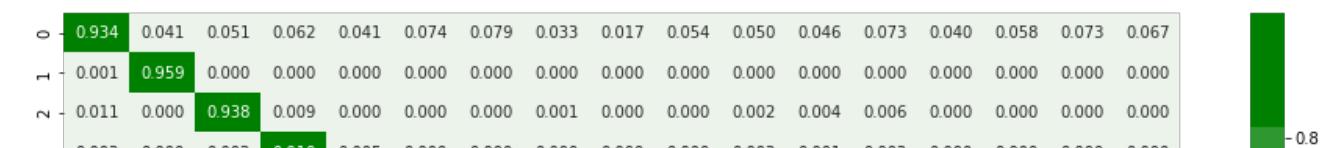
```
plot_confusion_matrix_2(y_val,y_hat_val)
```

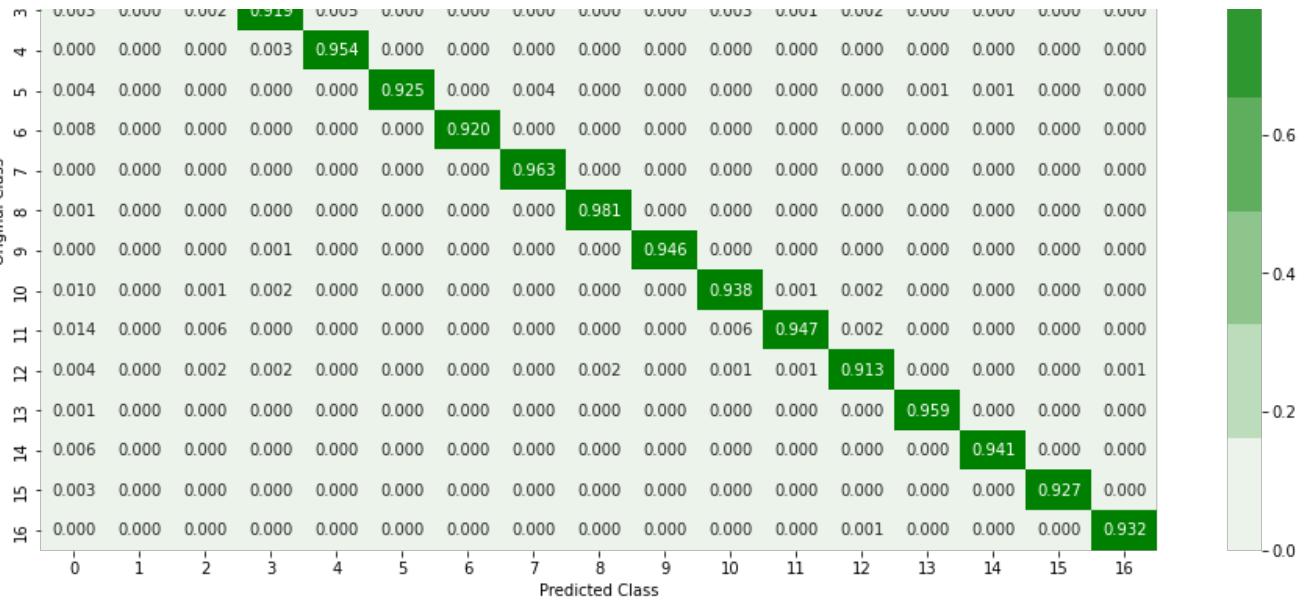
Confusion / Precision / Recall matrix
Percentage of misclassified points 6.329345703125

----- Confusion matrix -----



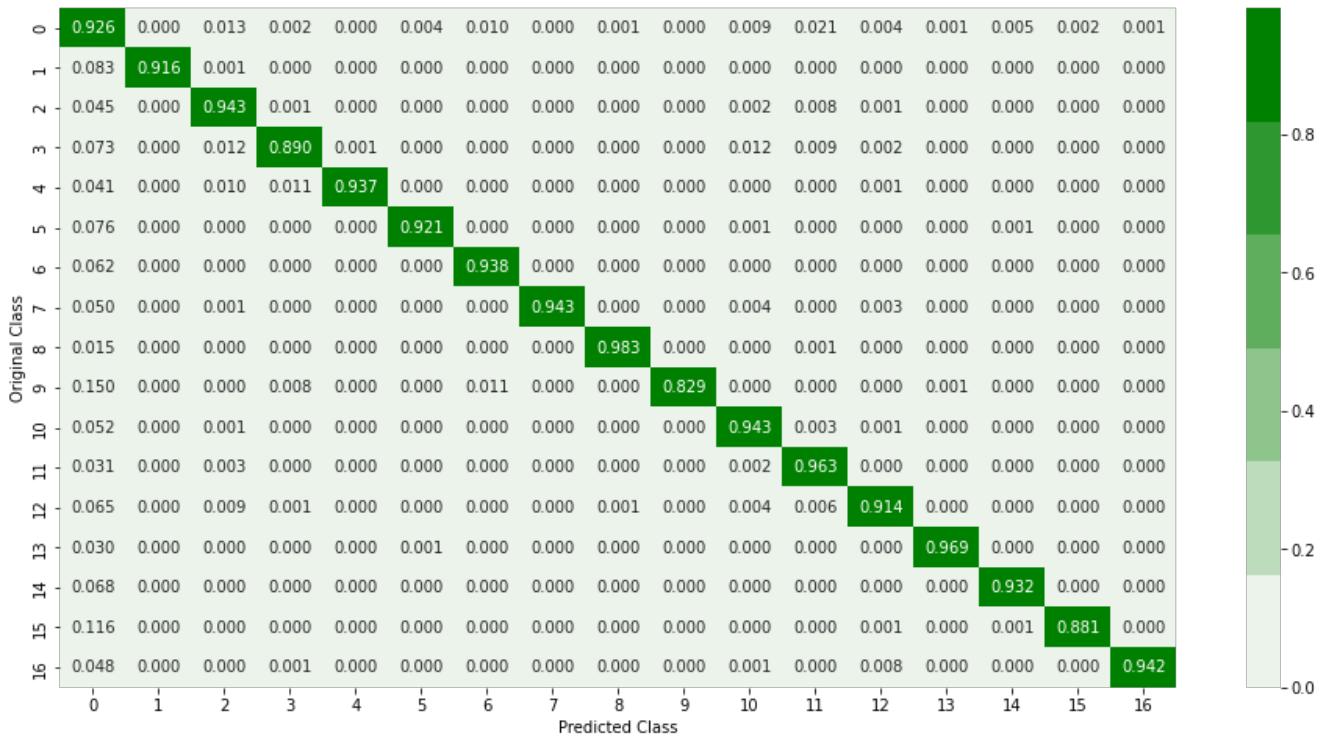
----- Precision matrix -----





Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

Recall matrix -----



Sum of rows in recall matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

micro F1 score : 0.93670654296875

Average Accuracy : 0.9277209705292182

In [74]:

```
F1_A2FPN_val = f1_score(y_val,y_hat_val,average='micro')
print('micro F1 score of pretrained unet model for validation data: ',F1_A2FPN_val)
kappa_A2FPN_val = cohen_kappa_score(y_val,y_hat_val)
print('kappa score of pretrained unet model for validation data: ',kappa_A2FPN_val)
```

micro F1 score of pretrained unet model for validation data: 0.93670654296875
kappa score of pretrained unet model for validation data: 0.917004769148751

In [75]:

```
# plt.figure(figsize=(15,15))
# im_count=1
# for i in range(10):
#     for j in range(10):
#         plt.subplot(10,10,im_count)
#         plt.imshow(img_pred[i][j])
#         im_count+=1
# plt.show()
```

Testing A2FPN model on unseen data

The score we see for the Full image segmentation is because the model has seen the class structures during the training. Its score drops for the validation set because it has some unseen data.

Point to be noted here is that the data of train and validation set comes from the same image patch with different augmentation.

The validation set will not have same image as training set but the regions of class within image will be shifted compared to the ones in train set. As the train/test split was generated from cropped images which have overlapping regions, most of the shapes of classes in the validation set are covered in train set except for few which reduced the score for validation set.

To know the true performance we need to Test the model on unseen data, where the class sizes are much different (smaller or bigger) compared to original image.

Since the only image we have here is 145 x 145, we shall construct image from the 64 x 64 images of test set. The new image will have the test set images overlapped on each other such that a 64 x 64 patch will have 4 (32 x 32) images. This will generate a New landscape where the classes do not have shapes same as the original Indian Pines. We shall extract the 64x64 patches from this newly generated image and test the model prediction.

In [76]:

```
# Selecting 64 x 64 images from test set to create new 145 x 145 image
test_image = X_test[::3]
test_image_gt = y_test[::3]
test_image.shape, test_image_gt.shape
```

Out[76]:

```
((67, 64, 64, 95), (67, 64, 64))
```

In [77]:

```
# for i in range(1):
#     figr,axis = plt.subplots(1,2,figsize=(10,10))
#     im0 = axis[0].imshow(test_image[2][:,:,20])#,cmap='jet')
#     axis[0].set_title('HSI')
#     plt.colorbar(im0,ax=axis[0],shrink=0.4,aspect=16), ticks=range(0,17,1))

#     im1 = axis[1].imshow(test_image_gt[2])#,cmap='jet')
#     axis[1].set_title('Ground Truth')
#     plt.colorbar(im1,ax=axis[1],shrink=0.4,aspect=16, ticks=range(0,17,1))
#     plt.show()
```

In [78]:

```
# 145 x 145 image generation
grid = 0
test_image_full = np.zeros((32*6, 32*6, 95))
test_image_gt_full = np.zeros((32*6, 32*6))
for i in range(5):
    for j in range(5):
        test_image_full[i*32:i*32+64,
                      j*32:j*32+64,:] = test_image[grid]
        test_image_gt_full[i*32:i*32+64,
                           j*32:j*32+64] = test_image_gt[grid]
    grid+=1

print('Test image size before cropping',test_image_full.shape, test_image_gt_full.shape)
```

```
test_image_full = test_image_full[0:145,0:145,:]
test_image_gt_full = test_image_gt_full[0:145,0:145]
print('Test image size after cropping',test_image_full.shape, test_image_gt_full.shape)
```

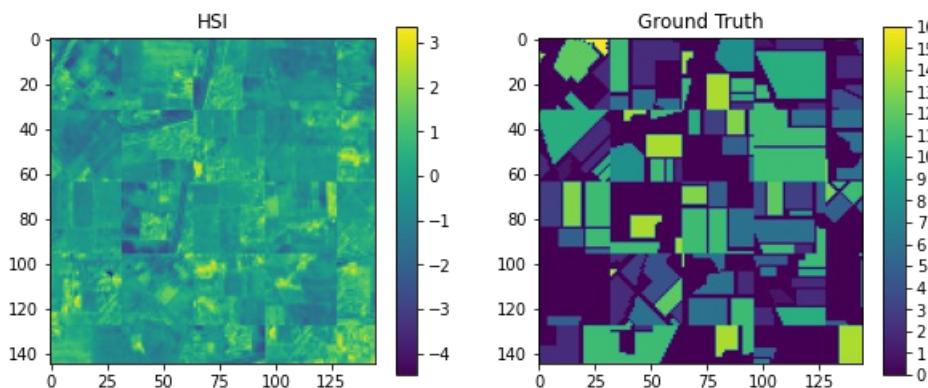
Test image size before cropping (192, 192, 95) (192, 192)
Test image size after cropping (145, 145, 95) (145, 145)

New Test Image

In [79]:

```
# New image
fig, axis = plt.subplots(1,2, figsize=(10,10))
im0 = axis[0].imshow(test_image_full[:, :, 30])#,cmap='jet')
axis[0].set_title('HSI')
plt.colorbar(im0, ax=axis[0], shrink=0.4, aspect=16) #, ticks=range(0,17,1))

im1 = axis[1].imshow(test_image_gt_full) #,cmap='jet')
axis[1].set_title('Ground Truth')
plt.colorbar(im1, ax=axis[1], shrink=0.4, aspect=16, ticks=range(0,17,1))
plt.show()
```



Generating patches for testing

In [80]:

```
# Generating the patches
test_img_pch = np.squeeze(patch.patchify(test_image_full, (64, 64, 95), step=9), axis=2)
test_img_gt_pch = patch.patchify(test_image_gt_full, (64, 64), step=9)
```

In [81]:

```
test_img_pch.shape, test_img_gt_pch.shape
```

Out[81]:

```
((10, 10, 64, 64, 95), (10, 10, 64, 64))
```

In [82]:

```
# Loading data associated with the new test image (145x145)
HSI_test_dataset = []
for i in range(test_img_pch.shape[0]):
    for j in range(test_img_pch.shape[1]):
        single_patch = test_img_pch[i][j]
        # single_patch = Std_scaler.transform(single_patch.reshape(-1,single_patch.shape[-1])).reshape(single_patch.shape)
        HSI_test_dataset.append(single_patch)
```

```
In [83]:
```

```
# Converting original patch list to numpy array
HSI_test_dataset = np.array(HSI_test_dataset)
```

```
In [84]:
```

```
# Generating Groundtruth dataset seperating the single 64x64 patch from patch grid (10,10,64,64)
HSI_test_gt_dataset = []
for i in range(test_img_gt_pch.shape[0]):
    for j in range(test_img_gt_pch.shape[1]):
        HSI_test_gt_dataset.append(patchs[i][j])
```

```
In [85]:
```

```
# Converting original gt patch list to numpy array
HSI_test_gt_dataset = np.array(HSI_test_gt_dataset)
```

```
In [86]:
```

```
# !wget --header="Host: doc-10-30-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Cookie: AUTH_82jcsesreiehjhkrct3c4mrj1raokod=00176583124175523585|1646480925000|qk60htbqipo58ucf58k773gpbn7i3tb" --header="Connection: keep-alive" "https://doc-10-30-docs.googleusercontent.com/docs/securescr90kivf62vcrm9d2s7vb24hsj0c3fo2/gasv5uimims8aemlsf4b0cpmk21o8ram/1646481150000/16522560826923149764/00176583124175523585/1JZ8M4EiQvHAyugt3qa3xUse9_XdTdID5?e=download&ax=ACxEAsZHpxRxDFeXGrXS2xu91-xukVMgCyRshdmr13n1o_XFFJkh3_XYJUZL13FK2wC49tRo50xxZtBUTzF11L4WasHFHVGhGOrA7jLsxVvGXCwIv6SALVopycpkf0btG_8ACBWmND2QQx8ONrex9HVIxbbaebI9P0IW2wSn_Thya0P2WtQ9x2p_prCeheOLG--mUszpSkiwb6GYSq07LWihqYWsAuqZJaCjHZhe6rD0maTwG03dsoLi0BsdZXzxWorX2qDEZhnoURWPzsXS9iMXAHoYPk2MBM55jfZLBKtjWk3fePgejXCjkLhj7FOeEsFD5CqnwpLzs_wvJ_oHiZ6vg_TceCiHcmQFvm2yYENeHtgfvwlRc-I1p41qdweQAOLX2RLCDO-ps-NwdzT1asvJm_hcu0H6MyrCJNtISTft7a5uvMC142_nmYX5Ur6joBrGT8-h5vrOYp51z2BhARCY5Q714nzbUEl1kEMpN7gWGeMnbEGuUftsAT5rl3p0o004TYdTGJaIT0qx_zbXXyo1hb1ocVBEtZTRUtnOGfmf6Nv0xf3xhACFZUcDafgzflw5lehHinFYqKO8ySScHqiVv2L5IZe0nFJpRoddO-ue-M7uGsakdqOFDKr7_dejNK1xFjDAFCXjTGHAluQ2oj4dQLOFV4hJ1x0Tj0YpeBoMZD&authuser=0" -c -O 'unet_m2_best_model_e50+49.h5'
```

```
In [87]:
```

```
# Loading saved model weights
# unet_m1.load_weights('unet_m1_best_model_e49.h5')
```

Model Prediction for the new test image patches

```
In [88]:
```

```
%%timeit
# predicting for individual patch
pred_test = FPN_model.predict(HSI_test_dataset)
```

```
1 loop, best of 5: 371 ms per loop
```

```
In [89]:
```

```
pred_test = FPN_model.predict(HSI_test_dataset)
```

```
In [90]:
```

```
pred_test.shape
```

```
Out[90]:
```

```
(100, 64, 64, 17)
```

Reconstructing the 145 x 145 image predictions

In [91]:

```
# Generating the 3D probabilities grid of all patches associated with full image.
grid = 0
grp = 0
img_prediction = np.zeros((145, 145, 100*17))
for i in range(10):
    for j in range(10):
        img_prediction[i*9:i*9+64,
                      j*9:j*9+64,
                      grp:grp+17] = pred_test[grid]
    grid+=1
    grp+=17

img_prediction.shape
```

Out[91]:

(145, 145, 1700)

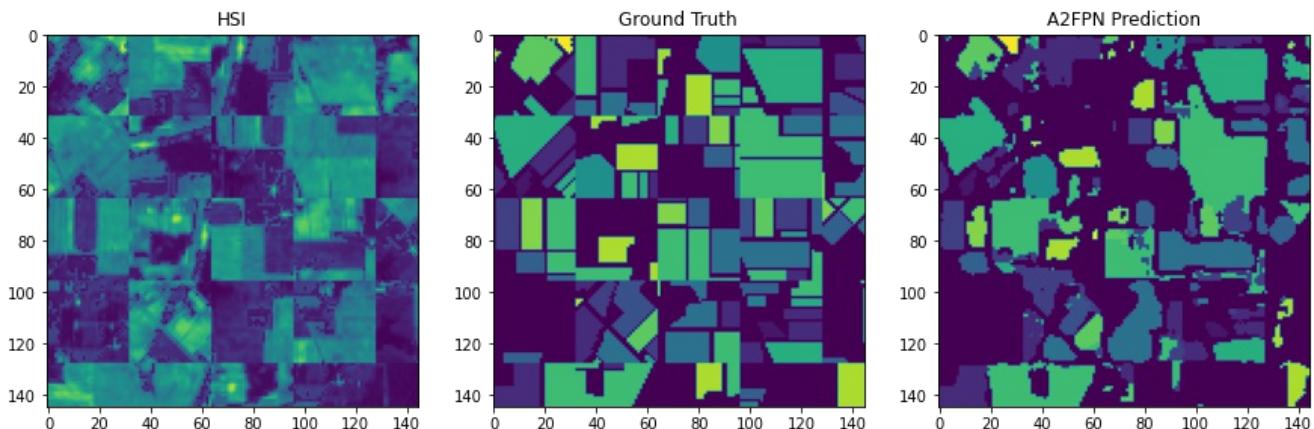
In [92]:

```
# Identifying the classes of each pixel from probabilities values of all patches corresponding to image
# (145x145)
prediction = np.argmax(img_prediction, axis=-1)%17
```

Prediction

In [93]:

```
# Plotting the segmentation after identifying the best class for overlapping patches
plt.figure(figsize=(15,15))
plt.subplot(131)
plt.imshow(test_image_full[:, :, 20])
plt.title('HSI')
plt.subplot(132)
plt.imshow(test_image_gt_full)
plt.title('Ground Truth')
plt.subplot(133)
plt.imshow(prediction)
plt.title('A2FPN Prediction')
plt.show()
```



Modified image prediction score (F1 and kappa)

In [94]:

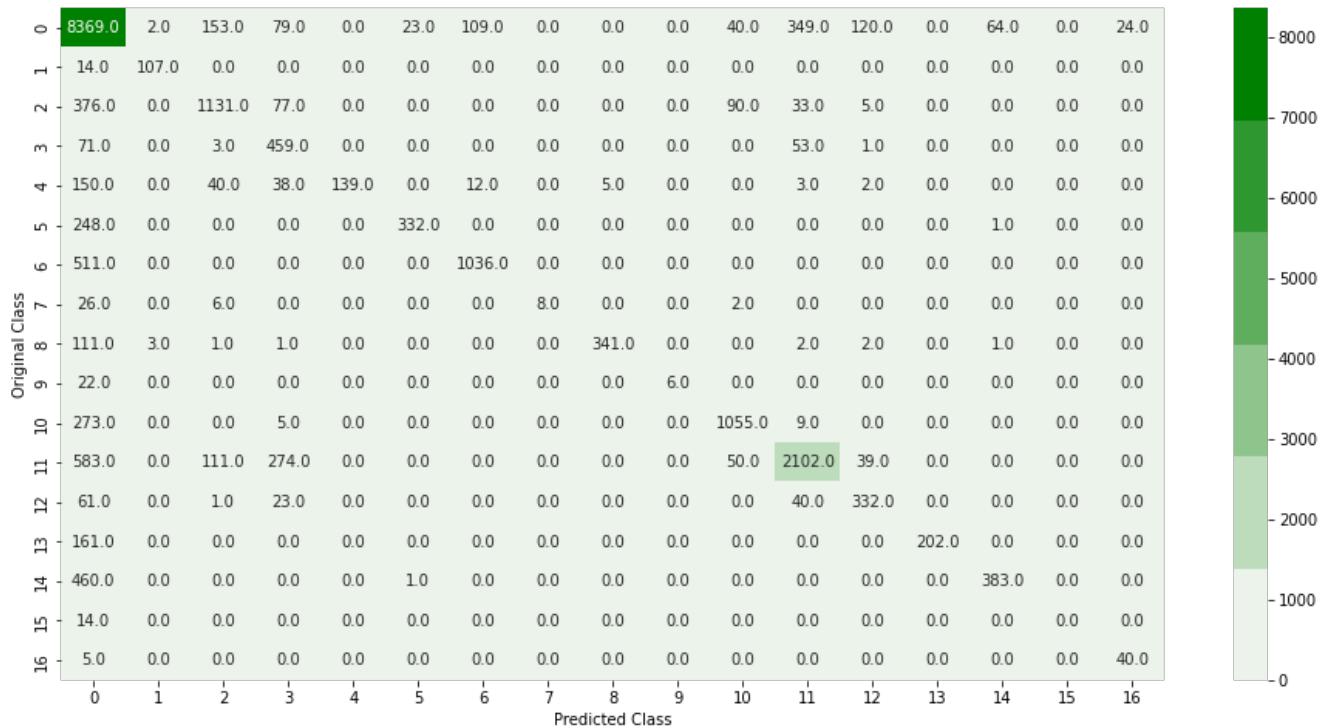
```
# Flattening the ground truths and predictions (145x145 image) for score evaluation
y = test_image_gt_full.flatten()
y_hat = prediction.flatten()
plot confusion matrix 2(v,v hat)
```

Confusion / Precision / Recall matrix

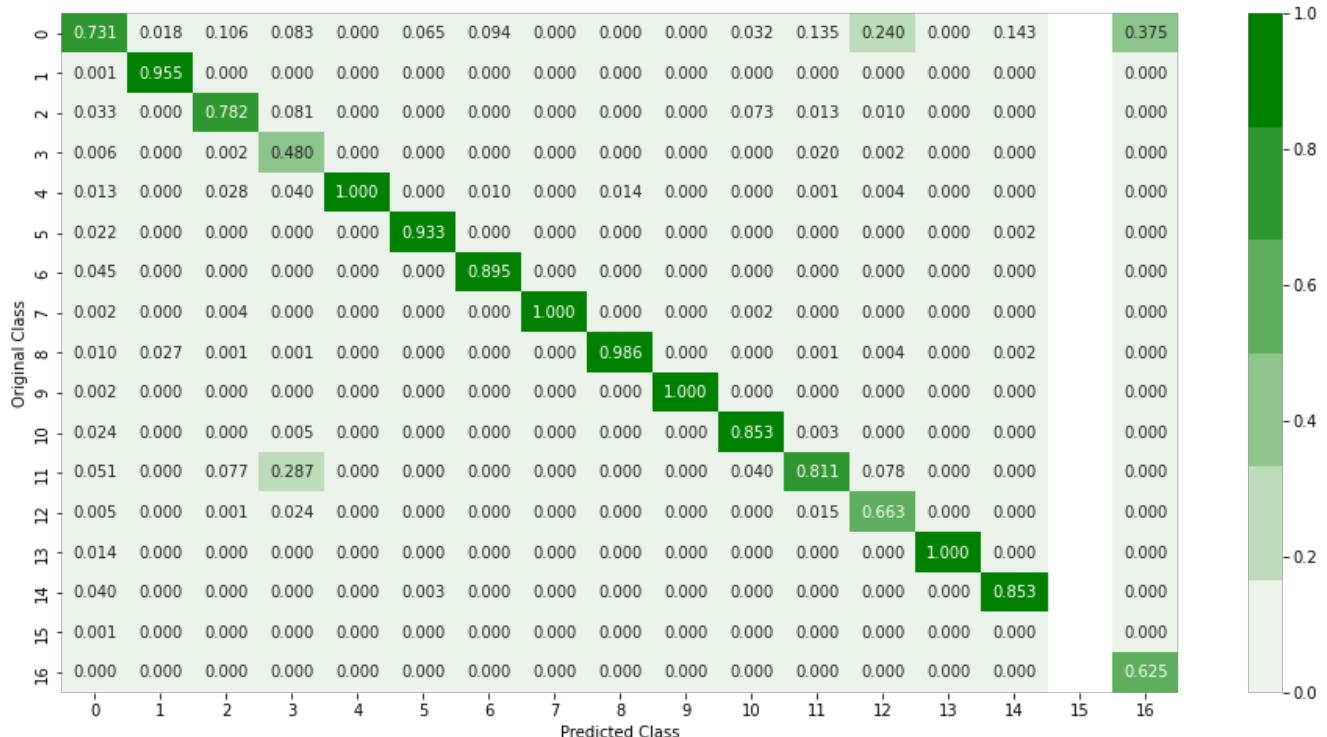
Percentage of misclassified points 23.700356718192626

----- Confusion matrix -----

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: RuntimeWarning: invalid value encountered in true_divide
```

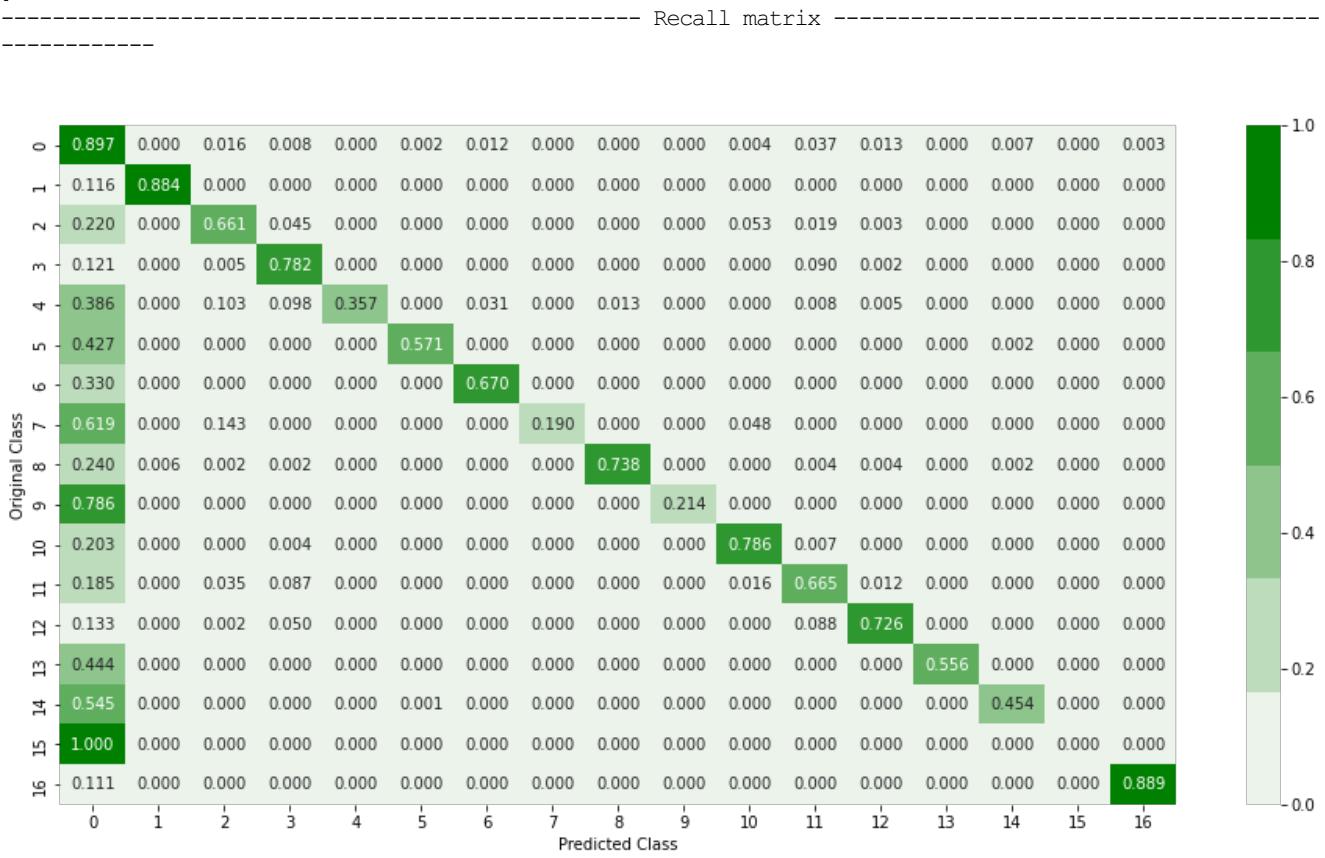


----- Precision matrix -----



Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. nan 1.

1



Sum of rows in recall matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

micro F1 score : 0.7629964328180737

Average Accuracy : 0.5907143718306725

Model is unable to identify and segment most of the classes. Most of them are classified under class 0.

In [96]:

```
F1_A2FPN = f1_score(y,y_hat,average='micro')
print('micro F1 score of pretrained unet model for full image : ',F1_A2FPN)
kappa_A2FPN = cohen_kappa_score(y,y_hat)
print('kappa score of pretrained unet model for full image : ',kappa_A2FPN)
```

micro F1 score of pretrained unet model for full image : 0.7629964328180737
kappa score of pretrained unet model for full image : 0.6719472781366087

Model 2 - Simple Unet

Here neither backbones nor pretrained weights are considered for Network architecture. Basic Unet model is constructed and trained from scratch for Indian Pines HSI data.

- The Encoder section of the network have convolutions with same padding settings and 3 levels of max pooling.
- The Decoder section of the has 3 levels of upconvolution operation where the upconv output are combined with the conv operation outputs of Encoder section.
- Output of Decoder network is passed through two stages of convolutions where final output is probabilities for 17 classes(64x64x17)

Model Definition

In []:

```
def simple_Unet(in_size,classes):
    '''This Function Generate and Returns Basic Unet model '''
    input = Input(in_size)
```

```

#Encoder Section
Enc_L1 = Conv2D(filters = 64, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(input)
Enc_L1 = Conv2D(filters = 64, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_L1)
Enc_P1 = MaxPooling2D(pool_size=(2, 2))(Enc_L1)

Enc_L2 = Conv2D(filters = 128, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_P1)
Enc_L2 = Conv2D(filters = 128, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_L2)
Enc_P2 = MaxPooling2D(pool_size=(2, 2))(Enc_L2)

Enc_L3 = Conv2D(filters = 256, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_P2)
Enc_L3 = Conv2D(filters = 256, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_L3)
Enc_P3 = MaxPooling2D(pool_size=(2, 2))(Enc_L3)

Enc_L4 = Conv2D(filters = 512, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_P3)
Enc_L4 = Conv2D(filters = 512, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_L4)
# Enc_P4 = MaxPooling2D(pool_size=(2, 2))(Enc_L4)

# Enc_L5 = Conv2D(filters = 1024, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_P4)
# Enc_L5 = Conv2D(filters = 1024, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Enc_L5)

# Dec_L0 = Conv2DTranspose(filters = 512, kernel_size = (2,2), strides =(2,2), padding='valid')(Enc_L5)
# Dec_L0 = concatenate([Dec_L0,Enc_L4])
# Dec_L0 = Conv2D(filters = 256, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Dec_L0)

# Decoder Section
Dec_L1 = Conv2DTranspose(filters = 256, kernel_size = (2,2), strides =(2,2), padding='valid')(Enc_L4)
Dec_L1 = concatenate([Dec_L1,Enc_L3])
Dec_L1 = Conv2D(filters = 256, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Dec_L1)

Dec_L2 = Conv2DTranspose(filters = 128, kernel_size = (2,2), strides =(2,2), padding='valid')(Dec_L1)
Dec_L2 = concatenate([Dec_L2,Enc_L2])
Dec_L2 = Conv2D(filters = 128, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Dec_L2)

Dec_L3 = Conv2DTranspose(filters = 64, kernel_size = (2,2), strides =(2,2), padding='valid')(Dec_L2)
Dec_L3 = concatenate([Dec_L3,Enc_L1])
Dec_L3 = Conv2D(filters = 64, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Dec_L3)

Dec_L4 = Conv2D(filters = 32, kernel_size = (3,3), padding='same', activation='relu',kernel_initializer='he_normal')(Dec_L3)

Output = Conv2D(filters = classes, kernel_size = (1,1), activation='softmax')(Dec_L4)

model = Model(inputs=input, outputs = Output)

return model

```

In []:

```
# del unet_m2
```

In []:

```
unet_m2 = simple_Unet((64,64,95),17)
```

In []:

```
unet_m2.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 64, 64, 95)]	0	[]
conv2d (Conv2D)	(None, 64, 64, 64)	54784	['input_1[0][0]']
conv2d_1 (Conv2D)	(None, 64, 64, 64)	36928	['conv2d[0][0]']
max_pooling2d (MaxPooling2D)	(None, 32, 32, 64)	0	['conv2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 32, 32, 128)	73856	['max_pooling2d[0][0]']
conv2d_3 (Conv2D)	(None, 32, 32, 128)	147584	['conv2d_2[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 128)	0	['conv2d_3[0][0]']
conv2d_4 (Conv2D)	(None, 16, 16, 256)	295168	['max_pooling2d_1[0][0]']
conv2d_5 (Conv2D)	(None, 16, 16, 256)	590080	['conv2d_4[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 256)	0	['conv2d_5[0][0]']
conv2d_6 (Conv2D)	(None, 8, 8, 512)	1180160	['max_pooling2d_2[0][0]']
conv2d_7 (Conv2D)	(None, 8, 8, 512)	2359808	['conv2d_6[0][0]']
conv2d_transpose (Conv2DTranspose)	(None, 16, 16, 256)	524544	['conv2d_7[0][0]']
concatenate (Concatenate)	(None, 16, 16, 512)	0	['conv2d_transpose[0][0]', 'conv2d_5[0][0]']
conv2d_8 (Conv2D)	(None, 16, 16, 256)	1179904	['concatenate[0][0]']
conv2d_transpose_1 (Conv2DTranspose)	(None, 32, 32, 128)	131200	['conv2d_8[0][0]']
concatenate_1 (Concatenate)	(None, 32, 32, 256)	0	['conv2d_transpose_1[0][0]', 'conv2d_3[0][0]']
conv2d_9 (Conv2D)	(None, 32, 32, 128)	295040	['concatenate_1[0][0]']
conv2d_transpose_2 (Conv2DTranspose)	(None, 64, 64, 64)	32832	['conv2d_9[0][0]']
concatenate_2 (Concatenate)	(None, 64, 64, 128)	0	['conv2d_transpose_2[0][0]', 'conv2d_1[0][0]']
conv2d_10 (Conv2D)	(None, 64, 64, 64)	73792	['concatenate_2[0][0]']
conv2d_11 (Conv2D)	(None, 64, 64, 32)	18464	['conv2d_10[0][0]']
conv2d_12 (Conv2D)	(None, 64, 64, 17)	561	['conv2d_11[0][0]']

Total params: 6,994,705

Trainable params: 6,994,705

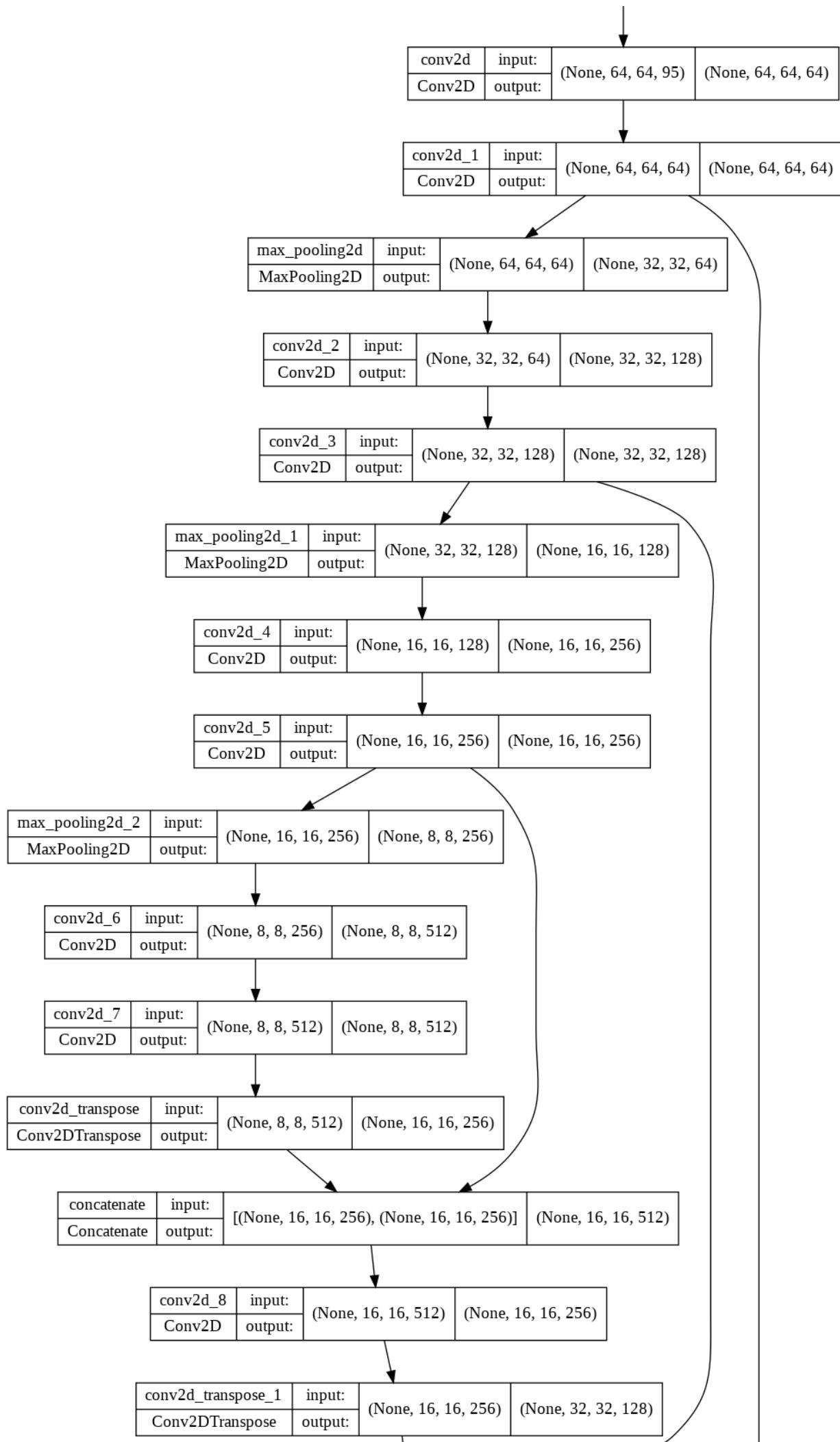
Non-trainable params: 0

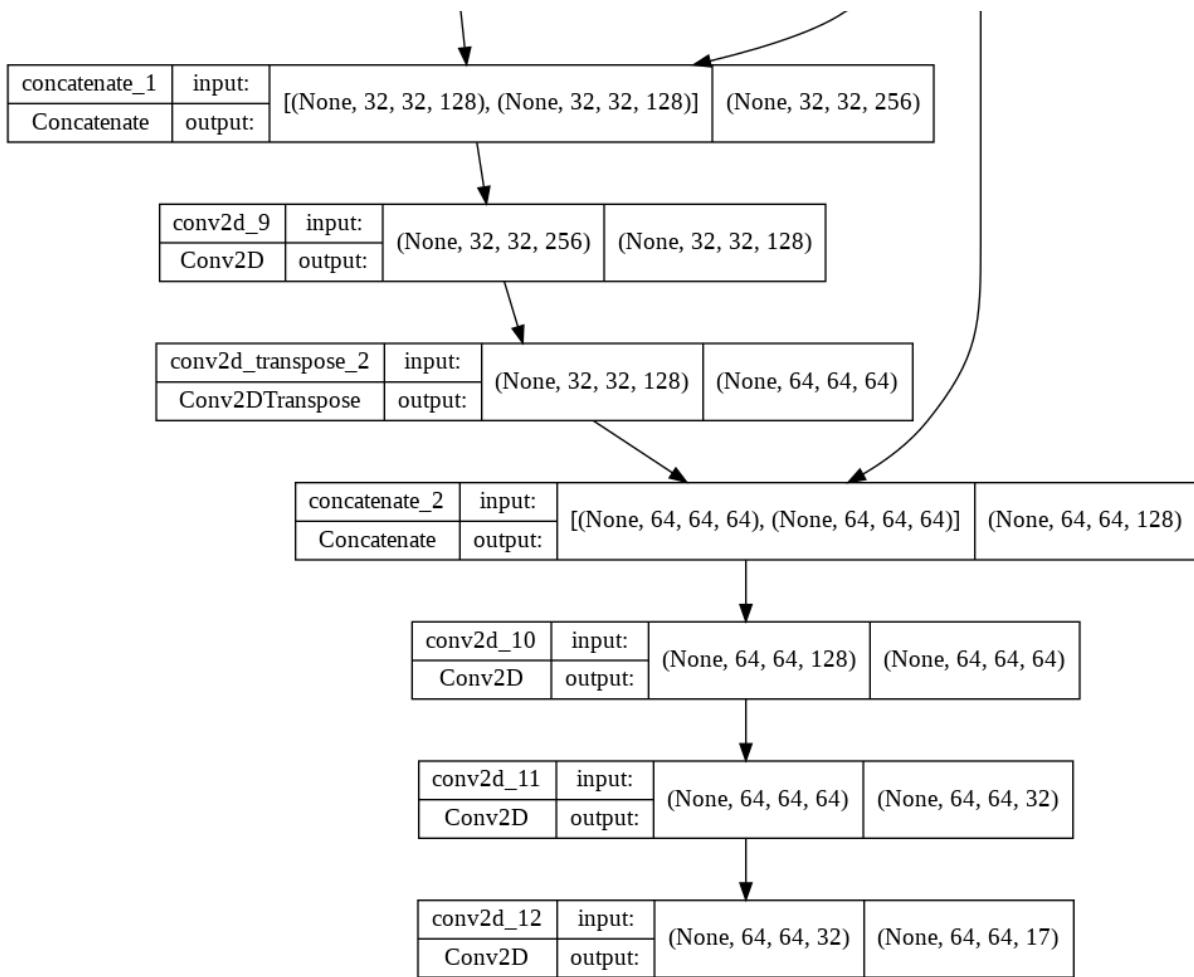
In []:

```
tf.keras.utils.plot_model(unet_m2, to_file='unet_m2.png', show_shapes=True, show_layer_names=True,  
rankdir='TB')
```

Out[]:

input_1	input:	[(None, 64, 64, 95)]	[(None, 64, 64, 95)]
InputLayer	output:		





Model Compile

In []:

```
optim = tf.keras.optimizers.Adam(0.0001)

focal_loss = sm.losses.cce_dice_loss #cce_dice_loss = categorical_crossentropy + dice_loss

unet_m2.compile(optim, focal_loss, metrics=[iou_score])
```

Model Training

Run 0

```
20220306-110614 WARNING:tensorflow:write_grads will be ignored in TensorFlow 2.0 for the TensorBoard Callback. Epoch 1/50 /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:20: UserWarning: Model.fit_generator is deprecated and will be removed in a future version. Please use Model.fit, which supports generators. 80/80 [=====] - ETA: 0s - loss: 0.9554 - iou_score: 0.0984 Epoch 1: val_iou_score improved from -inf to 0.18830, saving model to model_2_save/unet_m2_best_model_e01.h5 80/80 [=====] - 12s 125ms/step - loss: 0.9554 - iou_score: 0.0984 - val_loss: 0.8195 - val_iou_score: 0.1883 - lr: 1.0000e-04
```

```
Epoch 2/50 80/80 [=====] - ETA: 0s - loss: 0.6889 - iou_score: 0.2874 Epoch 2: val_iou_score improved from 0.18830 to 0.41579, saving model to model_2_save/unet_m2_best_model_e02.h5 80/80 [=====] - 9s 116ms/step - loss: 0.6889 - iou_score: 0.2874 - val_loss: 0.5426 - val_iou_score: 0.4158 - lr: 1.0000e-04
```

```
Epoch 3/50 80/80 [=====] - ETA: 0s - loss: 0.4978 - iou_score: 0.4501 Epoch 3: val_iou_score improved from 0.41579 to 0.50913, saving model to model_2_save/unet_m2_best_model_e03.h5 80/80 [=====] - 9s 116ms/step - loss: 0.4978 - iou_score: 0.4501 - val_loss: 0.4244 - val_iou_score: 0.5091 - lr: 1.0000e-04
```

```
Epoch 4/50 80/80 [=====] - ETA: 0s - loss: 0.3801 - iou_score: 0.5509 Epoch 4: val_iou_score
```

improved from 0.50913 to 0.62292, saving model to model_2_save/unet_m2_best_model_e04.h5 80/80
[=====] - 9s 115ms/step - loss: 0.3801 - iou_score: 0.5509 - val_loss: 0.3024 - val_iou_score: 0.6229 - lr: 1.0000e-04

Epoch 5/50 80/80 [=====] - ETA: 0s - loss: 0.2724 - iou_score: 0.6571 Epoch 5: val_iou_score improved from 0.62292 to 0.71145, saving model to model_2_save/unet_m2_best_model_e05.h5 80/80
[=====] - 9s 114ms/step - loss: 0.2724 - iou_score: 0.6571 - val_loss: 0.2171 - val_iou_score: 0.7114 - lr: 1.0000e-04

Epoch 6/50 80/80 [=====] - ETA: 0s - loss: 0.2079 - iou_score: 0.7242 Epoch 6: val_iou_score improved from 0.71145 to 0.75938, saving model to model_2_save/unet_m2_best_model_e06.h5

Epoch 6: ReduceLROnPlateau reducing learning rate to 8.999999772640876e-05. 80/80 [=====]
9s 114ms/step - loss: 0.2079 - iou_score: 0.7242 - val_loss: 0.1765 - val_iou_score: 0.7594 - lr: 1.0000e-04

Epoch 7/50 80/80 [=====] - ETA: 0s - loss: 0.1691 - iou_score: 0.7730 Epoch 7: val_iou_score improved from 0.75938 to 0.78585, saving model to model_2_save/unet_m2_best_model_e07.h5 80/80
[=====] - 9s 114ms/step - loss: 0.1691 - iou_score: 0.7730 - val_loss: 0.1584 - val_iou_score: 0.7859 - lr: 9.0000e-05

Epoch 8/50 80/80 [=====] - ETA: 0s - loss: 0.1507 - iou_score: 0.7977 Epoch 8: val_iou_score improved from 0.78585 to 0.80817, saving model to model_2_save/unet_m2_best_model_e08.h5 80/80
[=====] - 9s 114ms/step - loss: 0.1507 - iou_score: 0.7977 - val_loss: 0.1403 - val_iou_score: 0.8082 - lr: 9.0000e-05

Epoch 9/50 80/80 [=====] - ETA: 0s - loss: 0.1360 - iou_score: 0.8171 Epoch 9: val_iou_score improved from 0.80817 to 0.81239, saving model to model_2_save/unet_m2_best_model_e09.h5 80/80
[=====] - 9s 115ms/step - loss: 0.1360 - iou_score: 0.8171 - val_loss: 0.1373 - val_iou_score: 0.8124 - lr: 9.0000e-05

Epoch 10/50 80/80 [=====] - ETA: 0s - loss: 0.1261 - iou_score: 0.8303 Epoch 10: val_iou_score improved from 0.81239 to 0.82622, saving model to model_2_save/unet_m2_best_model_e10.h5 80/80
[=====] - 9s 115ms/step - loss: 0.1261 - iou_score: 0.8303 - val_loss: 0.1272 - val_iou_score: 0.8262 - lr: 9.0000e-05

Epoch 11/50 80/80 [=====] - ETA: 0s - loss: 0.1160 - iou_score: 0.8452 Epoch 11: val_iou_score improved from 0.82622 to 0.84190, saving model to model_2_save/unet_m2_best_model_e11.h5

Epoch 11: ReduceLROnPlateau reducing learning rate to 8.100000122794882e-05. 80/80 [=====]
- 9s 114ms/step - loss: 0.1160 - iou_score: 0.8452 - val_loss: 0.1159 - val_iou_score: 0.8419 - lr: 9.0000e-05

Epoch 12/50 80/80 [=====] - ETA: 0s - loss: 0.1056 - iou_score: 0.8599 Epoch 12: val_iou_score improved from 0.84190 to 0.85487, saving model to model_2_save/unet_m2_best_model_e12.h5 80/80
[=====] - 9s 114ms/step - loss: 0.1056 - iou_score: 0.8599 - val_loss: 0.1066 - val_iou_score: 0.8549 - lr: 8.1000e-05

Epoch 13/50 80/80 [=====] - ETA: 0s - loss: 0.0991 - iou_score: 0.8695 Epoch 13: val_iou_score improved from 0.85487 to 0.86230, saving model to model_2_save/unet_m2_best_model_e13.h5 80/80
[=====] - 9s 114ms/step - loss: 0.0991 - iou_score: 0.8695 - val_loss: 0.1013 - val_iou_score: 0.8623 - lr: 8.1000e-05

Epoch 14/50 80/80 [=====] - ETA: 0s - loss: 0.0952 - iou_score: 0.8754 Epoch 14: val_iou_score did not improve from 0.86230 80/80 [=====] - 9s 113ms/step - loss: 0.0952 - iou_score: 0.8754 - val_loss: 0.1048 - val_iou_score: 0.8580 - lr: 8.1000e-05

Epoch 15/50 80/80 [=====] - ETA: 0s - loss: 0.0950 - iou_score: 0.8756 Epoch 15: val_iou_score improved from 0.86230 to 0.86729, saving model to model_2_save/unet_m2_best_model_e15.h5 80/80
[=====] - 9s 115ms/step - loss: 0.0950 - iou_score: 0.8756 - val_loss: 0.0978 - val_iou_score: 0.8673 - lr: 8.1000e-05

Epoch 16/50 80/80 [=====] - ETA: 0s - loss: 0.0906 - iou_score: 0.8821 Epoch 16: val_iou_score improved from 0.86729 to 0.87348, saving model to model_2_save/unet_m2_best_model_e16.h5

Epoch 16: ReduceLROnPlateau reducing learning rate to 7.289999848580919e-05. 80/80 [=====]
- 9s 114ms/step - loss: 0.0906 - iou_score: 0.8821 - val_loss: 0.0941 - val_iou_score: 0.8735 - lr: 8.1000e-05

Epoch 17/50 80/80 [=====] - ETA: 0s - loss: 0.0845 - iou_score: 0.8916 Epoch 17: val_iou_score improved from 0.87348 to 0.87904, saving model to model_2_save/unet_m2_best_model_e17.h5 80/80
[=====] - 9s 114ms/step - loss: 0.0845 - iou_score: 0.8916 - val_loss: 0.0900 - val_iou_score: 0.8790 - lr: 7.2900e-05

Epoch 18/50 80/80 [=====] - ETA: 0s - loss: 0.0880 - iou_score: 0.8860 Epoch 18: val_iou_score did not improve from 0.87904 80/80 [=====] - 9s 114ms/step - loss: 0.0880 - iou_score: 0.8860 - val_loss: 0.0882 - val_iou_score: 0.8786 - lr: 7.2900e-05

Epoch 19/50 80/80 [=====] - ETA: 0s - loss: 0.0834 - iou_score: 0.8912 Epoch 19: val_iou_score improved from 0.87904 to 0.88355, saving model to model_2_save/unet_m2_best_model_e19.h5 80/80 [=====] - 9s 113ms/step - loss: 0.0834 - iou_score: 0.8912 - val_loss: 0.0857 - val_iou_score: 0.8836 - lr: 7.2900e-05

Epoch 20/50 80/80 [=====] - ETA: 0s - loss: 0.0792 - iou_score: 0.8977 Epoch 20: val_iou_score did not improve from 0.88355 80/80 [=====] - 9s 113ms/step - loss: 0.0792 - iou_score: 0.8977 - val_loss: 0.0936 - val_iou_score: 0.8723 - lr: 7.2900e-05

Epoch 21/50 80/80 [=====] - ETA: 0s - loss: 0.0812 - iou_score: 0.8942 Epoch 21: val_iou_score did not improve from 0.88355

Epoch 21: ReduceLROnPlateau reducing learning rate to 6.56100019114092e-05. 80/80 [=====] - 10s 130ms/step - loss: 0.0812 - iou_score: 0.8942 - val_loss: 0.0867 - val_iou_score: 0.8829 - lr: 7.2900e-05

Epoch 22/50 80/80 [=====] - ETA: 0s - loss: 0.0746 - iou_score: 0.9040 Epoch 22: val_iou_score improved from 0.88355 to 0.88714, saving model to model_2_save/unet_m2_best_model_e22.h5 80/80 [=====] - 11s 143ms/step - loss: 0.0746 - iou_score: 0.9040 - val_loss: 0.0833 - val_iou_score: 0.8871 - lr: 6.5610e-05

Epoch 23/50 80/80 [=====] - ETA: 0s - loss: 0.0697 - iou_score: 0.9110 Epoch 23: val_iou_score improved from 0.88714 to 0.89337, saving model to model_2_save/unet_m2_best_model_e23.h5 80/80 [=====] - 10s 119ms/step - loss: 0.0697 - iou_score: 0.9110 - val_loss: 0.0795 - val_iou_score: 0.8934 - lr: 6.5610e-05

Epoch 24/50 80/80 [=====] - ETA: 0s - loss: 0.0669 - iou_score: 0.9152 Epoch 24: val_iou_score improved from 0.89337 to 0.89623, saving model to model_2_save/unet_m2_best_model_e24.h5 80/80 [=====] - 9s 115ms/step - loss: 0.0669 - iou_score: 0.9152 - val_loss: 0.0765 - val_iou_score: 0.8962 - lr: 6.5610e-05

Epoch 25/50 80/80 [=====] - ETA: 0s - loss: 0.0654 - iou_score: 0.9174 Epoch 25: val_iou_score improved from 0.89623 to 0.89789, saving model to model_2_save/unet_m2_best_model_e25.h5 80/80 [=====] - 9s 115ms/step - loss: 0.0654 - iou_score: 0.9174 - val_loss: 0.0755 - val_iou_score: 0.8979 - lr: 6.5610e-05

Epoch 26/50 80/80 [=====] - ETA: 0s - loss: 0.0624 - iou_score: 0.9219 Epoch 26: val_iou_score improved from 0.89789 to 0.90072, saving model to model_2_save/unet_m2_best_model_e26.h5

Epoch 26: ReduceLROnPlateau reducing learning rate to 5.904900172026828e-05. 80/80 [=====] - 9s 115ms/step - loss: 0.0624 - iou_score: 0.9219 - val_loss: 0.0734 - val_iou_score: 0.9007 - lr: 6.5610e-05

Epoch 27/50 80/80 [=====] - ETA: 0s - loss: 0.0612 - iou_score: 0.9237 Epoch 27: val_iou_score improved from 0.90072 to 0.90142, saving model to model_2_save/unet_m2_best_model_e27.h5 80/80 [=====] - 9s 115ms/step - loss: 0.0612 - iou_score: 0.9237 - val_loss: 0.0728 - val_iou_score: 0.9014 - lr: 5.9049e-05

Epoch 28/50 80/80 [=====] - ETA: 0s - loss: 0.0605 - iou_score: 0.9248 Epoch 28: val_iou_score improved from 0.90142 to 0.90302, saving model to model_2_save/unet_m2_best_model_e28.h5 80/80 [=====] - 11s 138ms/step - loss: 0.0605 - iou_score: 0.9248 - val_loss: 0.0724 - val_iou_score: 0.9030 - lr: 5.9049e-05

Epoch 29/50 80/80 [=====] - ETA: 0s - loss: 0.0637 - iou_score: 0.9201 Epoch 29: val_iou_score improved from 0.90302 to 0.90589, saving model to model_2_save/unet_m2_best_model_e29.h5 80/80 [=====] - 9s 115ms/step - loss: 0.0637 - iou_score: 0.9201 - val_loss: 0.0700 - val_iou_score: 0.9059 - lr: 5.9049e-05

Epoch 30/50 80/80 [=====] - ETA: 0s - loss: 0.0607 - iou_score: 0.9245 Epoch 30: val_iou_score did not improve from 0.90589 80/80 [=====] - 10s 123ms/step - loss: 0.0607 - iou_score: 0.9245 - val_loss: 0.0761 - val_iou_score: 0.8976 - lr: 5.9049e-05

Epoch 31/50 80/80 [=====] - ETA: 0s - loss: 0.0596 - iou_score: 0.9260 Epoch 31: val_iou_score improved from 0.90589 to 0.90886, saving model to model_2_save/unet_m2_best_model_e31.h5

Epoch 31: ReduceLROnPlateau reducing learning rate to 5.314410154824145e-05. 80/80 [=====] - 11s 137ms/step - loss: 0.0596 - iou_score: 0.9260 - val_loss: 0.0679 - val_iou_score: 0.9089 - lr: 5.9049e-05

Epoch 32/50 80/80 [=====] - ETA: 0s - loss: 0.0559 - iou_score: 0.9316 Epoch 32: val_iou_score improved from 0.90886 to 0.90907, saving model to model_2_save/unet_m2_best_model_e32.h5 80/80
[=====] - 10s 127ms/step - loss: 0.0559 - iou_score: 0.9316 - val_loss: 0.0679 - val_iou_score: 0.9091 - lr: 5.3144e-05

Epoch 33/50 80/80 [=====] - ETA: 0s - loss: 0.0563 - iou_score: 0.9310 Epoch 33: val_iou_score improved from 0.90907 to 0.91024, saving model to model_2_save/unet_m2_best_model_e33.h5 80/80
[=====] - 9s 114ms/step - loss: 0.0563 - iou_score: 0.9310 - val_loss: 0.0670 - val_iou_score: 0.9102 - lr: 5.3144e-05

Epoch 34/50 80/80 [=====] - ETA: 0s - loss: 0.0554 - iou_score: 0.9309 Epoch 34: val_iou_score did not improve from 0.91024 80/80 [=====] - 10s 122ms/step - loss: 0.0554 - iou_score: 0.9309 - val_loss: 0.0809 - val_iou_score: 0.8931 - lr: 5.3144e-05

Epoch 35/50 80/80 [=====] - ETA: 0s - loss: 0.0611 - iou_score: 0.9217 Epoch 35: val_iou_score did not improve from 0.91024 80/80 [=====] - 9s 114ms/step - loss: 0.0611 - iou_score: 0.9217 - val_loss: 0.1142 - val_iou_score: 0.8488 - lr: 5.3144e-05

Epoch 36/50 80/80 [=====] - ETA: 0s - loss: 0.0725 - iou_score: 0.9040 Epoch 36: val_iou_score did not improve from 0.91024

Epoch 36: ReduceLROnPlateau reducing learning rate to 4.7829690083744934e-05. 80/80
[=====] - 9s 114ms/step - loss: 0.0725 - iou_score: 0.9040 - val_loss: 0.0811 - val_iou_score: 0.8910 - lr: 5.3144e-05

Epoch 37/50 80/80 [=====] - ETA: 0s - loss: 0.0507 - iou_score: 0.9337 Epoch 37: val_iou_score did not improve from 0.91024 80/80 [=====] - 9s 114ms/step - loss: 0.0507 - iou_score: 0.9337 - val_loss: 0.0698 - val_iou_score: 0.9064 - lr: 4.7830e-05

Epoch 38/50 80/80 [=====] - ETA: 0s - loss: 0.0445 - iou_score: 0.9426 Epoch 38: val_iou_score did not improve from 0.91024 80/80 [=====] - 10s 128ms/step - loss: 0.0445 - iou_score: 0.9426 - val_loss: 0.0863 - val_iou_score: 0.8850 - lr: 4.7830e-05

Epoch 39/50 80/80 [=====] - ETA: 0s - loss: 0.0496 - iou_score: 0.9346 Epoch 39: val_iou_score improved from 0.91024 to 0.91262, saving model to model_2_save/unet_m2_best_model_e39.h5 80/80
[=====] - 9s 116ms/step - loss: 0.0496 - iou_score: 0.9346 - val_loss: 0.0656 - val_iou_score: 0.9126 - lr: 4.7830e-05

Epoch 40/50 80/80 [=====] - ETA: 0s - loss: 0.0560 - iou_score: 0.9258 Epoch 40: val_iou_score did not improve from 0.91262 80/80 [=====] - 9s 113ms/step - loss: 0.0560 - iou_score: 0.9258 - val_loss: 0.0691 - val_iou_score: 0.9076 - lr: 4.7830e-05

Epoch 41/50 80/80 [=====] - ETA: 0s - loss: 0.0411 - iou_score: 0.9461 Epoch 41: val_iou_score did not improve from 0.91262

Epoch 41: ReduceLROnPlateau reducing learning rate to 4.304672074795235e-05. 80/80 [=====]
- 9s 113ms/step - loss: 0.0411 - iou_score: 0.9461 - val_loss: 0.0655 - val_iou_score: 0.9115 - lr: 4.7830e-05

Epoch 42/50 80/80 [=====] - ETA: 0s - loss: 0.0547 - iou_score: 0.9254 Epoch 42: val_iou_score did not improve from 0.91262 80/80 [=====] - 9s 113ms/step - loss: 0.0547 - iou_score: 0.9254 - val_loss: 0.0688 - val_iou_score: 0.9064 - lr: 4.3047e-05

Epoch 43/50 80/80 [=====] - ETA: 0s - loss: 0.0431 - iou_score: 0.9412 Epoch 43: val_iou_score did not improve from 0.91262 80/80 [=====] - 9s 113ms/step - loss: 0.0431 - iou_score: 0.9412 - val_loss: 0.1746 - val_iou_score: 0.7986 - lr: 4.3047e-05

Epoch 44/50 80/80 [=====] - ETA: 0s - loss: 0.1135 - iou_score: 0.8533 Epoch 44: val_iou_score did not improve from 0.91262 80/80 [=====] - 9s 113ms/step - loss: 0.1135 - iou_score: 0.8533 - val_loss: 0.0779 - val_iou_score: 0.8926 - lr: 4.3047e-05

Epoch 45/50 80/80 [=====] - ETA: 0s - loss: 0.0508 - iou_score: 0.9280 Epoch 45: val_iou_score did not improve from 0.91262 80/80 [=====] - 9s 113ms/step - loss: 0.0508 - iou_score: 0.9280 - val_loss: 0.0663 - val_iou_score: 0.9090 - lr: 4.3047e-05

Epoch 46/50 80/80 [=====] - ETA: 0s - loss: 0.0320 - iou_score: 0.9534 Epoch 46: val_iou_score improved from 0.91262 to 0.91766, saving model to model_2_save/unet_m2_best_model_e46.h5

Epoch 46: ReduceLROnPlateau reducing learning rate to 3.8742047036066654e-05. 80/80
[=====] - 9s 116ms/step - loss: 0.0320 - iou_score: 0.9534 - val_loss: 0.0599 - val_iou_score: 0.9177 - lr: 4.3047e-05 Epoch 47/50

```
80/80 [=====] - ETA: 0s - loss: 0.0295 - iou_score: 0.9567 Epoch 47: val_iou_score did not improve from 0.91766 80/80 [=====] - 9s 115ms/step - loss: 0.0295 - iou_score: 0.9567 - val_loss: 0.0609 - val_iou_score: 0.9169 - lr: 3.8742e-05
```

```
Epoch 48/50 80/80 [=====] - ETA: 0s - loss: 0.0273 - iou_score: 0.9595 Epoch 48: val_iou_score did not improve from 0.91766 80/80 [=====] - 9s 114ms/step - loss: 0.0273 - iou_score: 0.9595 - val_loss: 0.0611 - val_iou_score: 0.9162 - lr: 3.8742e-05
```

```
Epoch 49/50 80/80 [=====] - ETA: 0s - loss: 0.0233 - iou_score: 0.9652 Epoch 49: val_iou_score improved from 0.91766 to 0.92356, saving model to model_2_save/unet_m2_best_model_e49.h5 80/80 [=====] - 9s 117ms/step - loss: 0.0233 - iou_score: 0.9652 - val_loss: 0.0561 - val_iou_score: 0.9236 - lr: 3.8742e-05
```

```
Epoch 50/50 80/80 [=====] - ETA: 0s - loss: 0.0211 - iou_score: 0.9686 Epoch 50: val_iou_score improved from 0.92356 to 0.92518, saving model to model_2_save/unet_m2_best_model_e50.h5 80/80 [=====] - 9s 116ms/step - loss: 0.0211 - iou_score: 0.9686 - val_loss: 0.0549 - val_iou_score: 0.9252 - lr: 3.8742e-05 Time Taken for training (sec): 483.12275314331055
```

Run 1

In []:

```
# loading model weights from 50th epoch
unet_m2.load_weights('/content/model_2_save/unet_m2_best_model_e50.h5')
```

In []:

```
#lr 3.8742e-05 at 50 epoch
optim = tf.keras.optimizers.Adam(3.8742e-05)

focal_loss = sm.losses.cce_dice_loss #cce_dice_loss = categorical_crossentropy + dice_loss
unet_m2.compile(optim, focal_loss, metrics=[iou_score])
```

In []:

```
datetime_stamp = datetime.now().strftime("%Y%m%d-%H%M%S")
logdir = os.path.join("logs", datetime_stamp)
print(datetime_stamp)
# tensorboard = TensorBoard(log_dir=logdir)
tensorboard = TensorBoard(log_dir=logdir, histogram_freq=1, write_graph=True, write_grads=True)

checkpoint_m2 = ModelCheckpoint('model_2_save2/unet_m2_best_model_e{epoch:02d}.h5',
                               save_weights_only=True, save_best_only=True, mode='max',
                               monitor='val_iou_score', verbose=1)

Reduce_LR_m2 = ReduceLROnPlateau(monitor='val_iou_score', factor=0.9, min_lr=0.00001, patience=5, verbose=1)

callbacks_m2 = [checkpoint_m2, Reduce_LR_m2, tensorboard]

start = time.time()
history_m2 = unet_m2.fit_generator(train_dataloader,
                                     steps_per_epoch=len(train_dataloader),
                                     epochs=50,
                                     validation_data=test_dataloader,
                                     callbacks=callbacks_m2)

stop = time.time()
print('Time Taken for training (sec): ', stop-start)
```

20220306-114109

WARNING:tensorflow: `write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/50

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:20: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

00/00 [=====] - ETA: 0s - loss: 0.0229 - iou_score: 0.9660

```
ou/ou [=====] - ETA: 0s - loss: 0.0229 - iou_score: 0.9660
Epoch 1: val_iou_score improved from -inf to 0.92713, saving model to model_2_save2/unet_m2_best_model_e01.h5
80/80 [=====] - 11s 121ms/step - loss: 0.0229 - iou_score: 0.9660 - val_loss: 0.0532 - val_iou_score: 0.9271 - lr: 3.8742e-05
Epoch 2/50
80/80 [=====] - ETA: 0s - loss: 0.0202 - iou_score: 0.9694
Epoch 2: val_iou_score did not improve from 0.92713
80/80 [=====] - 10s 121ms/step - loss: 0.0202 - iou_score: 0.9694 - val_loss: 0.0564 - val_iou_score: 0.9218 - lr: 3.8742e-05
Epoch 3/50
80/80 [=====] - ETA: 0s - loss: 0.0210 - iou_score: 0.9683
Epoch 3: val_iou_score improved from 0.92713 to 0.92752, saving model to model_2_save2/unet_m2_best_model_e03.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0210 - iou_score: 0.9683 - val_loss: 0.0531 - val_iou_score: 0.9275 - lr: 3.8742e-05
Epoch 4/50
80/80 [=====] - ETA: 0s - loss: 0.0173 - iou_score: 0.9737
Epoch 4: val_iou_score improved from 0.92752 to 0.93106, saving model to model_2_save2/unet_m2_best_model_e04.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0173 - iou_score: 0.9737 - val_loss: 0.0505 - val_iou_score: 0.9311 - lr: 3.8742e-05
Epoch 5/50
80/80 [=====] - ETA: 0s - loss: 0.0164 - iou_score: 0.9751
Epoch 5: val_iou_score improved from 0.93106 to 0.93121, saving model to model_2_save2/unet_m2_best_model_e05.h5
80/80 [=====] - 9s 114ms/step - loss: 0.0164 - iou_score: 0.9751 - val_loss: 0.0505 - val_iou_score: 0.9312 - lr: 3.8742e-05
Epoch 6/50
80/80 [=====] - ETA: 0s - loss: 0.0156 - iou_score: 0.9763
Epoch 6: val_iou_score improved from 0.93121 to 0.93167, saving model to model_2_save2/unet_m2_best_model_e06.h5
80/80 [=====] - 9s 114ms/step - loss: 0.0156 - iou_score: 0.9763 - val_loss: 0.0502 - val_iou_score: 0.9317 - lr: 3.8742e-05
Epoch 7/50
80/80 [=====] - ETA: 0s - loss: 0.0149 - iou_score: 0.9773
Epoch 7: val_iou_score did not improve from 0.93167

Epoch 7: ReduceLROnPlateau reducing learning rate to 3.486780042294413e-05.
80/80 [=====] - 9s 113ms/step - loss: 0.0149 - iou_score: 0.9773 - val_loss: 0.0506 - val_iou_score: 0.9311 - lr: 3.8742e-05
Epoch 8/50
80/80 [=====] - ETA: 0s - loss: 0.0145 - iou_score: 0.9780
Epoch 8: val_iou_score improved from 0.93167 to 0.93283, saving model to model_2_save2/unet_m2_best_model_e08.h5
80/80 [=====] - 9s 117ms/step - loss: 0.0145 - iou_score: 0.9780 - val_loss: 0.0492 - val_iou_score: 0.9328 - lr: 3.4868e-05
Epoch 9/50
80/80 [=====] - ETA: 0s - loss: 0.0141 - iou_score: 0.9785
Epoch 9: val_iou_score improved from 0.93283 to 0.93315, saving model to model_2_save2/unet_m2_best_model_e09.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0141 - iou_score: 0.9785 - val_loss: 0.0492 - val_iou_score: 0.9331 - lr: 3.4868e-05
Epoch 10/50
80/80 [=====] - ETA: 0s - loss: 0.0135 - iou_score: 0.9794
Epoch 10: val_iou_score did not improve from 0.93315
80/80 [=====] - 9s 113ms/step - loss: 0.0135 - iou_score: 0.9794 - val_loss: 0.0491 - val_iou_score: 0.9330 - lr: 3.4868e-05
Epoch 11/50
80/80 [=====] - ETA: 0s - loss: 0.0137 - iou_score: 0.9790
Epoch 11: val_iou_score improved from 0.93315 to 0.93360, saving model to model_2_save2/unet_m2_best_model_e11.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0137 - iou_score: 0.9790 - val_loss: 0.0489 - val_iou_score: 0.9336 - lr: 3.4868e-05
Epoch 12/50
80/80 [=====] - ETA: 0s - loss: 0.0131 - iou_score: 0.9800
Epoch 12: val_iou_score improved from 0.93360 to 0.93510, saving model to model_2_save2/unet_m2_best_model_e12.h5

Epoch 12: ReduceLROnPlateau reducing learning rate to 3.138102038064972e-05.
80/80 [=====] - 9s 114ms/step - loss: 0.0131 - iou_score: 0.9800 - val_loss: 0.0475 - val_iou_score: 0.9351 - lr: 3.4868e-05
Epoch 13/50
80/80 [=====] - ETA: 0s - loss: 0.0125 - iou_score: 0.9809
Epoch 13: val_iou_score did not improve from 0.93510
80/80 [=====] - 9s 113ms/step - loss: 0.0125 - iou_score: 0.9809 - val_loss: 0.0483 - val_iou_score: 0.9346 - lr: 3.1381e-05
11/50
```

```
Epoch 14/50
80/80 [=====] - ETA: 0s - loss: 0.0120 - iou_score: 0.9816
Epoch 14: val_iou_score improved from 0.93510 to 0.93522, saving model to model_2_save2/unet_m2_best_mo
del_e14.h5
80/80 [=====] - 9s 117ms/step - loss: 0.0120 - iou_score: 0.9816 - val_loss: 0
.0475 - val_iou_score: 0.9352 - lr: 3.1381e-05
Epoch 15/50
80/80 [=====] - ETA: 0s - loss: 0.0121 - iou_score: 0.9815
Epoch 15: val_iou_score did not improve from 0.93522
80/80 [=====] - 9s 113ms/step - loss: 0.0121 - iou_score: 0.9815 - val_loss: 0
.0487 - val_iou_score: 0.9342 - lr: 3.1381e-05
Epoch 16/50
80/80 [=====] - ETA: 0s - loss: 0.0116 - iou_score: 0.9821
Epoch 16: val_iou_score did not improve from 0.93522
80/80 [=====] - 10s 121ms/step - loss: 0.0116 - iou_score: 0.9821 - val_loss:
0.0484 - val_iou_score: 0.9349 - lr: 3.1381e-05
Epoch 17/50
80/80 [=====] - ETA: 0s - loss: 0.0111 - iou_score: 0.9829
Epoch 17: val_iou_score did not improve from 0.93522

Epoch 17: ReduceLROnPlateau reducing learning rate to 2.824291768774856e-05.
80/80 [=====] - 9s 113ms/step - loss: 0.0111 - iou_score: 0.9829 - val_loss: 0
.0495 - val_iou_score: 0.9336 - lr: 3.1381e-05
Epoch 18/50
80/80 [=====] - ETA: 0s - loss: 0.0109 - iou_score: 0.9831
Epoch 18: val_iou_score did not improve from 0.93522
80/80 [=====] - 9s 113ms/step - loss: 0.0109 - iou_score: 0.9831 - val_loss: 0
.0491 - val_iou_score: 0.9344 - lr: 2.8243e-05
Epoch 19/50
80/80 [=====] - ETA: 0s - loss: 0.0104 - iou_score: 0.9840
Epoch 19: val_iou_score improved from 0.93522 to 0.93655, saving model to model_2_save2/unet_m2_best_mo
del_e19.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0104 - iou_score: 0.9840 - val_loss: 0
.0469 - val_iou_score: 0.9365 - lr: 2.8243e-05
Epoch 20/50
80/80 [=====] - ETA: 0s - loss: 0.0102 - iou_score: 0.9842
Epoch 20: val_iou_score did not improve from 0.93655
80/80 [=====] - 9s 114ms/step - loss: 0.0102 - iou_score: 0.9842 - val_loss: 0
.0470 - val_iou_score: 0.9364 - lr: 2.8243e-05
Epoch 21/50
80/80 [=====] - ETA: 0s - loss: 0.0101 - iou_score: 0.9844
Epoch 21: val_iou_score did not improve from 0.93655
80/80 [=====] - 9s 114ms/step - loss: 0.0101 - iou_score: 0.9844 - val_loss: 0
.0469 - val_iou_score: 0.9365 - lr: 2.8243e-05
Epoch 22/50
80/80 [=====] - ETA: 0s - loss: 0.0113 - iou_score: 0.9824
Epoch 22: val_iou_score did not improve from 0.93655

Epoch 22: ReduceLROnPlateau reducing learning rate to 2.5418625591555612e-05.
80/80 [=====] - 9s 115ms/step - loss: 0.0113 - iou_score: 0.9824 - val_loss: 0
.0531 - val_iou_score: 0.9274 - lr: 2.8243e-05
Epoch 23/50
80/80 [=====] - ETA: 0s - loss: 0.0122 - iou_score: 0.9814
Epoch 23: val_iou_score improved from 0.93655 to 0.93694, saving model to model_2_save2/unet_m2_best_mo
del_e23.h5
80/80 [=====] - 9s 116ms/step - loss: 0.0122 - iou_score: 0.9814 - val_loss: 0
.0463 - val_iou_score: 0.9369 - lr: 2.5419e-05
Epoch 24/50
80/80 [=====] - ETA: 0s - loss: 0.0100 - iou_score: 0.9845
Epoch 24: val_iou_score did not improve from 0.93694
80/80 [=====] - 9s 114ms/step - loss: 0.0100 - iou_score: 0.9845 - val_loss: 0
.0478 - val_iou_score: 0.9351 - lr: 2.5419e-05
Epoch 25/50
80/80 [=====] - ETA: 0s - loss: 0.0103 - iou_score: 0.9840
Epoch 25: val_iou_score improved from 0.93694 to 0.93838, saving model to model_2_save2/unet_m2_best_mo
del_e25.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0103 - iou_score: 0.9840 - val_loss: 0
.0456 - val_iou_score: 0.9384 - lr: 2.5419e-05
Epoch 26/50
80/80 [=====] - ETA: 0s - loss: 0.0094 - iou_score: 0.9855
Epoch 26: val_iou_score improved from 0.93838 to 0.93897, saving model to model_2_save2/unet_m2_best_mo
del_e26.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0094 - iou_score: 0.9855 - val_loss: 0
.0449 - val_iou_score: 0.9390 - lr: 2.5419e-05
Epoch 27/50
80/80 [=====] - ETA: 0s - loss: 0.0092 - iou_score: 0.9858
Epoch 27: val_iou_score did not improve from 0.93897
```

```
Epoch 27: ReduceLROnPlateau reducing learning rate to 2.2876762704981958e-05.
80/80 [=====] - 9s 114ms/step - loss: 0.0092 - iou_score: 0.9858 - val_loss: 0
.0451 - val_iou_score: 0.9387 - lr: 2.5419e-05
Epoch 28/50
80/80 [=====] - ETA: 0s - loss: 0.0089 - iou_score: 0.9862
Epoch 28: val_iou_score did not improve from 0.93897
80/80 [=====] - 9s 114ms/step - loss: 0.0089 - iou_score: 0.9862 - val_loss: 0
.0455 - val_iou_score: 0.9384 - lr: 2.2877e-05
Epoch 29/50
80/80 [=====] - ETA: 0s - loss: 0.0085 - iou_score: 0.9869
Epoch 29: val_iou_score did not improve from 0.93897
80/80 [=====] - 9s 114ms/step - loss: 0.0085 - iou_score: 0.9869 - val_loss: 0
.0455 - val_iou_score: 0.9386 - lr: 2.2877e-05
Epoch 30/50
80/80 [=====] - ETA: 0s - loss: 0.0084 - iou_score: 0.9870
Epoch 30: val_iou_score improved from 0.93897 to 0.93960, saving model to model_2_save2/unet_m2_best_mo
del_e30.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0084 - iou_score: 0.9870 - val_loss: 0
.0444 - val_iou_score: 0.9396 - lr: 2.2877e-05
Epoch 31/50
80/80 [=====] - ETA: 0s - loss: 0.0081 - iou_score: 0.9875
Epoch 31: val_iou_score improved from 0.93960 to 0.93985, saving model to model_2_save2/unet_m2_best_mo
del_e31.h5
80/80 [=====] - 9s 116ms/step - loss: 0.0081 - iou_score: 0.9875 - val_loss: 0
.0444 - val_iou_score: 0.9398 - lr: 2.2877e-05
Epoch 32/50
80/80 [=====] - ETA: 0s - loss: 0.0079 - iou_score: 0.9878
Epoch 32: val_iou_score did not improve from 0.93985

Epoch 32: ReduceLROnPlateau reducing learning rate to 2.0589085943356624e-05.
80/80 [=====] - 9s 115ms/step - loss: 0.0079 - iou_score: 0.9878 - val_loss: 0
.0444 - val_iou_score: 0.9397 - lr: 2.2877e-05
Epoch 33/50
80/80 [=====] - ETA: 0s - loss: 0.0077 - iou_score: 0.9881
Epoch 33: val_iou_score did not improve from 0.93985
80/80 [=====] - 9s 113ms/step - loss: 0.0077 - iou_score: 0.9881 - val_loss: 0
.0447 - val_iou_score: 0.9395 - lr: 2.0589e-05
Epoch 34/50
80/80 [=====] - ETA: 0s - loss: 0.0075 - iou_score: 0.9884
Epoch 34: val_iou_score improved from 0.93985 to 0.94074, saving model to model_2_save2/unet_m2_best_mo
del_e34.h5
80/80 [=====] - 9s 117ms/step - loss: 0.0075 - iou_score: 0.9884 - val_loss: 0
.0435 - val_iou_score: 0.9407 - lr: 2.0589e-05
Epoch 35/50
80/80 [=====] - ETA: 0s - loss: 0.0074 - iou_score: 0.9885
Epoch 35: val_iou_score did not improve from 0.94074
80/80 [=====] - 9s 114ms/step - loss: 0.0074 - iou_score: 0.9885 - val_loss: 0
.0446 - val_iou_score: 0.9396 - lr: 2.0589e-05
Epoch 36/50
80/80 [=====] - ETA: 0s - loss: 0.0074 - iou_score: 0.9886
Epoch 36: val_iou_score did not improve from 0.94074
80/80 [=====] - 9s 115ms/step - loss: 0.0074 - iou_score: 0.9886 - val_loss: 0
.0441 - val_iou_score: 0.9402 - lr: 2.0589e-05
Epoch 37/50
80/80 [=====] - ETA: 0s - loss: 0.0072 - iou_score: 0.9888
Epoch 37: val_iou_score did not improve from 0.94074

Epoch 37: ReduceLROnPlateau reducing learning rate to 1.85301778401481e-05.
80/80 [=====] - 9s 114ms/step - loss: 0.0072 - iou_score: 0.9888 - val_loss: 0
.0443 - val_iou_score: 0.9400 - lr: 2.0589e-05
Epoch 38/50
80/80 [=====] - ETA: 0s - loss: 0.0071 - iou_score: 0.9890
Epoch 38: val_iou_score did not improve from 0.94074
80/80 [=====] - 9s 114ms/step - loss: 0.0071 - iou_score: 0.9890 - val_loss: 0
.0441 - val_iou_score: 0.9401 - lr: 1.8530e-05
Epoch 39/50
80/80 [=====] - ETA: 0s - loss: 0.0070 - iou_score: 0.9892
Epoch 39: val_iou_score improved from 0.94074 to 0.94161, saving model to model_2_save2/unet_m2_best_mo
del_e39.h5
80/80 [=====] - 9s 116ms/step - loss: 0.0070 - iou_score: 0.9892 - val_loss: 0
.0428 - val_iou_score: 0.9416 - lr: 1.8530e-05
Epoch 40/50
80/80 [=====] - ETA: 0s - loss: 0.0069 - iou_score: 0.9893
Epoch 40: val_iou_score did not improve from 0.94161
80/80 [=====] - 9s 115ms/step - loss: 0.0069 - iou_score: 0.9893 - val_loss: 0
.0436 - val_iou_score: 0.9406 - lr: 1.8530e-05
```

```

Epoch 41/50
80/80 [=====] - ETA: 0s - loss: 0.0068 - iou_score: 0.9894
Epoch 41: val_iou_score did not improve from 0.94161
80/80 [=====] - 9s 113ms/step - loss: 0.0068 - iou_score: 0.9894 - val_loss: 0
.0430 - val_iou_score: 0.9414 - lr: 1.8530e-05
Epoch 42/50
80/80 [=====] - ETA: 0s - loss: 0.0067 - iou_score: 0.9896
Epoch 42: val_iou_score improved from 0.94161 to 0.94197, saving model to model_2_save2/unet_m2_best_mo
del_e42.h5

Epoch 42: ReduceLROnPlateau reducing learning rate to 1.667716005613329e-05.
80/80 [=====] - 9s 115ms/step - loss: 0.0067 - iou_score: 0.9896 - val_loss: 0
.0425 - val_iou_score: 0.9420 - lr: 1.8530e-05
Epoch 43/50
80/80 [=====] - ETA: 0s - loss: 0.0066 - iou_score: 0.9897
Epoch 43: val_iou_score did not improve from 0.94197
80/80 [=====] - 9s 113ms/step - loss: 0.0066 - iou_score: 0.9897 - val_loss: 0
.0427 - val_iou_score: 0.9419 - lr: 1.6677e-05
Epoch 44/50
80/80 [=====] - ETA: 0s - loss: 0.0065 - iou_score: 0.9898
Epoch 44: val_iou_score improved from 0.94197 to 0.94199, saving model to model_2_save2/unet_m2_best_mo
del_e44.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0065 - iou_score: 0.9898 - val_loss: 0
.0427 - val_iou_score: 0.9420 - lr: 1.6677e-05
Epoch 45/50
80/80 [=====] - ETA: 0s - loss: 0.0064 - iou_score: 0.9900
Epoch 45: val_iou_score improved from 0.94199 to 0.94222, saving model to model_2_save2/unet_m2_best_mo
del_e45.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0064 - iou_score: 0.9900 - val_loss: 0
.0424 - val_iou_score: 0.9422 - lr: 1.6677e-05
Epoch 46/50
80/80 [=====] - ETA: 0s - loss: 0.0064 - iou_score: 0.9900
Epoch 46: val_iou_score did not improve from 0.94222
80/80 [=====] - 9s 114ms/step - loss: 0.0064 - iou_score: 0.9900 - val_loss: 0
.0432 - val_iou_score: 0.9413 - lr: 1.6677e-05
Epoch 47/50
80/80 [=====] - ETA: 0s - loss: 0.0064 - iou_score: 0.9901
Epoch 47: val_iou_score improved from 0.94222 to 0.94246, saving model to model_2_save2/unet_m2_best_mo
del_e47.h5

Epoch 47: ReduceLROnPlateau reducing learning rate to 1.50094445416471e-05.
80/80 [=====] - 9s 116ms/step - loss: 0.0064 - iou_score: 0.9901 - val_loss: 0
.0420 - val_iou_score: 0.9425 - lr: 1.6677e-05
Epoch 48/50
80/80 [=====] - ETA: 0s - loss: 0.0063 - iou_score: 0.9901
Epoch 48: val_iou_score did not improve from 0.94246
80/80 [=====] - 9s 114ms/step - loss: 0.0063 - iou_score: 0.9901 - val_loss: 0
.0425 - val_iou_score: 0.9421 - lr: 1.5009e-05
Epoch 49/50
80/80 [=====] - ETA: 0s - loss: 0.0062 - iou_score: 0.9903
Epoch 49: val_iou_score improved from 0.94246 to 0.94278, saving model to model_2_save2/unet_m2_best_mo
del_e49.h5
80/80 [=====] - 9s 115ms/step - loss: 0.0062 - iou_score: 0.9903 - val_loss: 0
.0419 - val_iou_score: 0.9428 - lr: 1.5009e-05
Epoch 50/50
80/80 [=====] - ETA: 0s - loss: 0.0062 - iou_score: 0.9902
Epoch 50: val_iou_score did not improve from 0.94278
80/80 [=====] - 9s 114ms/step - loss: 0.0062 - iou_score: 0.9902 - val_loss: 0
.0429 - val_iou_score: 0.9416 - lr: 1.5009e-05
Time Taken for training (sec): 462.4833571910858

```

In []:

```
# # http://localhost:6006/
%load_ext tensorboard
%tensorboard --logdir logs --host localhost
```

In []:

```
# index of max iou_score
np.argmax(history_m2.history['val_iou_score'])
```

Out[]:

Predicting patches using Best unet_m2 weights

In []:

```
unet_m2.load_weights('/content/model_2_save2/unet_m2_best_model_e49.h5')
```

In []:

```
!wget --header="Host: doc-14-5s-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Cookie: AUTH_7ss66jgs9bhkt1fs3a2o48dsvbafhp10=16522560826923149764|1646630925000|qanl2uccok319urr9t5hd91138k b7gsk" --header="Connection: keep-alive" "https://doc-14-5s-docs.googleusercontent.com/docs/securesc/qoko4v8vsugpnd5ekm0hgah4cdscvg6u/ou3vr1uu0mfogg9vr3ds6vs8di16f6h/1646631075000/00176583124175523585/16522560826923149764/1bzB6PQYUF6S62HVeH3fB44z2-p55bwyt?e=download&ax=ACxEAsaKbzTk45pmbS2yzHykp1EvHfi4YSWj2pZHEKdTm5hPf5KZ7Z0eAoRY8TP6K-mH7SmgrvNr_wpH5hmgNDRg7Qm-r3pAeVCju6TvemNY1aJREJZmvkB2Y_zGFu37LTjX1q7r0ixvD WQf3Y-s24pGKnCCBmCvB40bmQbCVjnpIPrZTrIs1hb-4wR3Frki2GOPIj-TcXY45xU-egfb2b87mQk_z1WX_f8iz8L0m8k2eMOxCIJC LzrIm8F3XU9P6VmHu74xi8m0mJI6osbmU4N05ju8gIjomPsdvSPK29BPpfSMocrVVjyxyXKRJ5BCTdLRLbdHzKry6apde3BCxttV3ye JOzbFwuZnPGmNz8ZQXZZY5yw1xuRxJ_vhGyNcxNYC1yv8w094zd_uThsgeNX_AoZrza55gzg3eHoFGKAE9aLbt4JMgSxCE-Vb69nXm6 etzqc9v2BAYeN3d6oVPOwHHIufD8wT7CwhQyPq05t-yRoF6GBeBW1FK2aOKn5M_N0JOjQAy8J6ZTwIREHL_AJFBb-VODO5dPedb1V0I YBrRMkhBURd7GOWqc5gauwD0hfleIacnxjcrkLu8TIBEkuz5MY4AfPsdkn0eJqlQDuUlG_BHVqNZ-eD14kN42X8VNpfLIHVuaSVWYvm Mjs3YyN4_gAoTkGeFIHylIFKc-B&authuser=0" -c -O 'unet_m2_best_model_e50+49.h5'
```

```
--2022-03-07 05:32:47-- https://doc-14-5s-docs.googleusercontent.com/docs/securesc/qoko4v8vsugpnd5ekm0hgah4cdscvg6u/ou3vr1uu0mfogg9vr3ds6vs8di16f6h/1646631075000/00176583124175523585/16522560826923149764/1bzB6PQYUF6S62HVeH3fB44z2-p55bwyt?e=download&ax=ACxEAsaKbzTk45pmbS2yzHykp1EvHfi4YSWj2pZHEKdTm5hPf5KZ7Z0eAoRY8TP6K-mH7SmgrvNr_wpH5hmgNDRg7Qm-r3pAeVCju6TvemNY1aJREJZmvkB2Y_zGFu37LTjX1q7r0ixvD WQf3Y-s24pGKnCCBmCvB40bmQbCVjnpIPrZTrIs1hb-4wR3Frki2GOPIj-TcXY45xU-egfb2b87mQk_z1WX_f8iz8L0m8k2eMOxCIJC LzrIm8F3XU9P6VmHu74xi8m0mJI6osbmU4N05ju8gIjomPsdvSPK29BPpfSMocrVVjyxyXKRJ5BCTdLRLbdHzKry6apde3BCxttV3ye JOzbFwuZnPGmNz8ZQXZZY5yw1xuRxJ_vhGyNcxNYC1yv8w094zd_uThsgeNX_AoZrza55gzg3eHoFGKAE9aLbt4JMgSxCE-Vb69nXm6 etzqc9v2BAYeN3d6oVPOwHHIufD8wT7CwhQyPq05t-yRoF6GBeBW1FK2aOKn5M_N0JOjQAy8J6ZTwIREHL_AJFBb-VODO5dPedb1V0IYBrRMkhBURd7GOWqc5gauwD0hfleIacnxjcrkLu8TIBEkuz5MY4AfPsdkn0eJqlQDuUlG_BHVqNZ-eD14kN42X8VNpfLIHVuaSVWYvmMjs3YyN4_gAoTkGeFIHylIFKc-B&authuser=0
Resolving doc-14-5s-docs.googleusercontent.com (doc-14-5s-docs.googleusercontent.com) ... 108.177.125.132, 2404:6800:4008:c01::84
Connecting to doc-14-5s-docs.googleusercontent.com (doc-14-5s-docs.googleusercontent.com) |108.177.125.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28038648 (27M) [application/octet-stream]
Saving to: 'unet_m2_best_model_e50+49.h5'

unet_m2_best_model_ 100%[=====] 26.74M 163MB/s in 0.2s

2022-03-07 05:32:48 (163 MB/s) - 'unet_m2_best_model_e50+49.h5' saved [28038648/28038648]
```

In []:

```
# Loading saved model weights
unet_m2.load_weights('unet_m2_best_model_e50+49.h5')
```

In []:

```
# Plotting Model prediction of segmentation alongside HSI and Ground Truth
i=0
for im, gt in zip(X_test[20:100],y_test[20:100]):

    # model prediction
    pred = unet_m2.predict(im[np.newaxis,:,:,:])

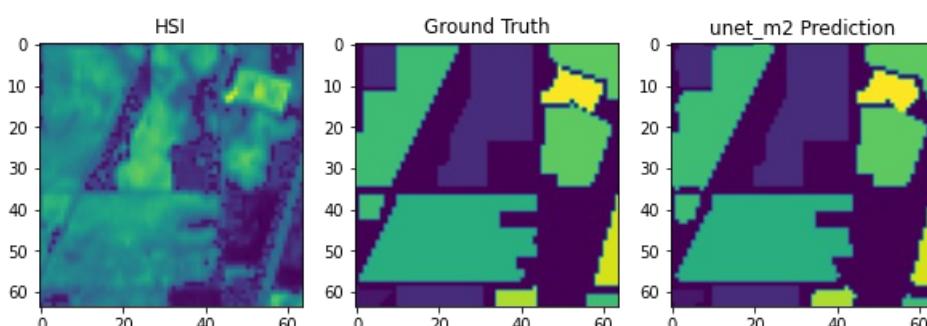
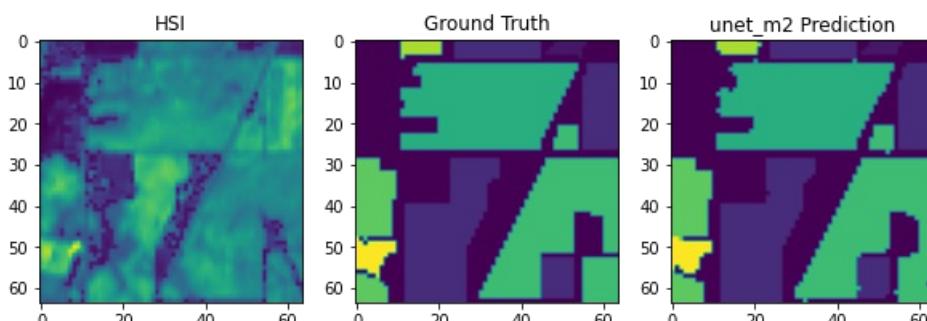
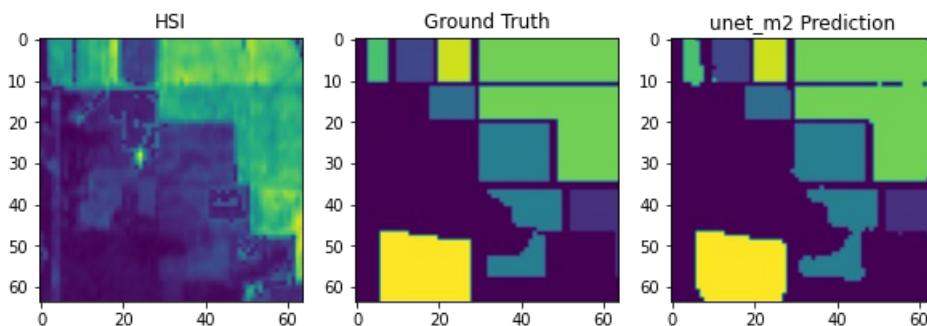
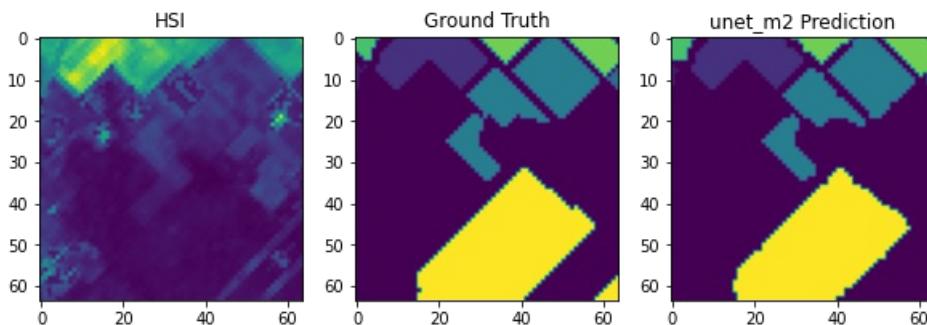
    # generating the image based on the max probability of particular class
    prediction = np.argmax(pred, axis=-1)

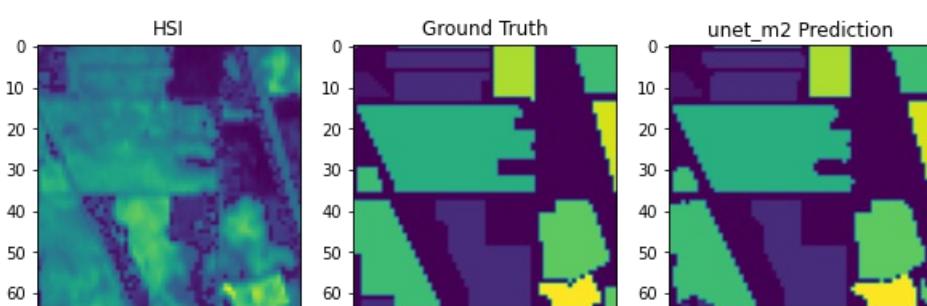
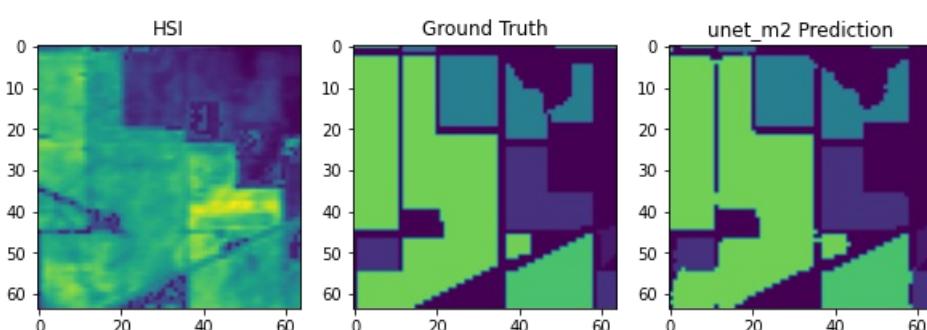
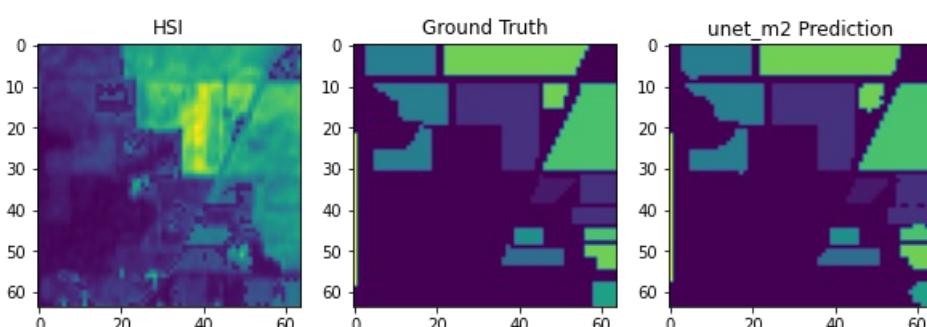
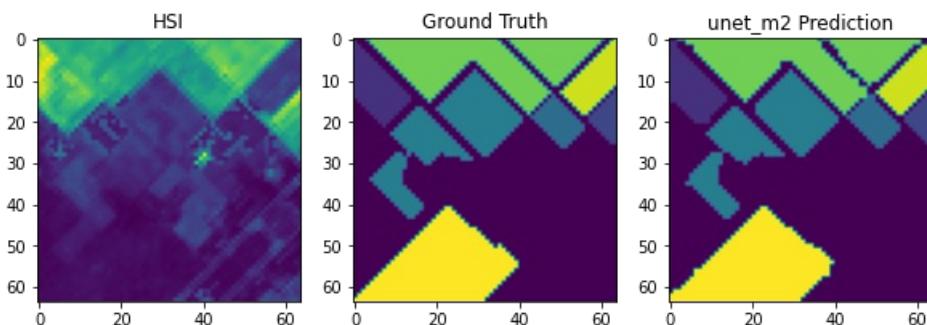
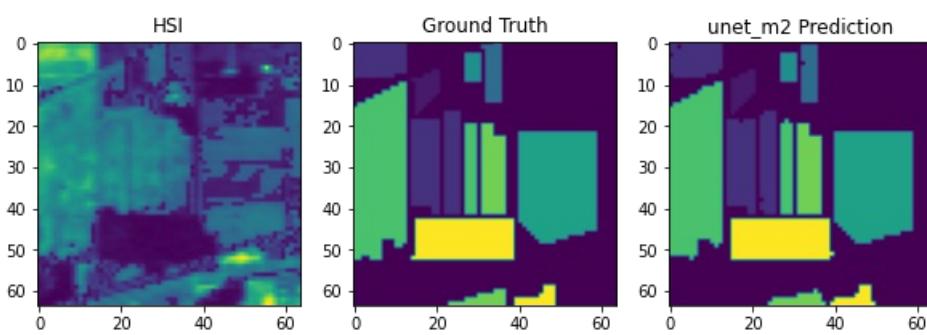
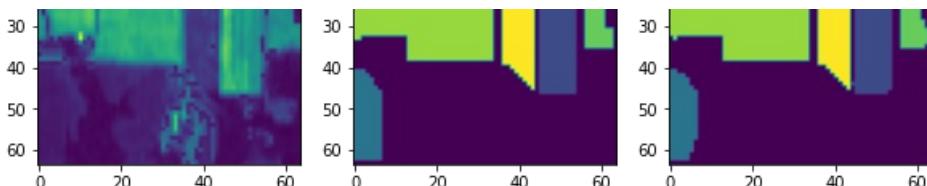
    # plotting HSI image vs ground truth vs prediction
    plt.figure(figsize=(10,6))
    plt.subplot(131)
```

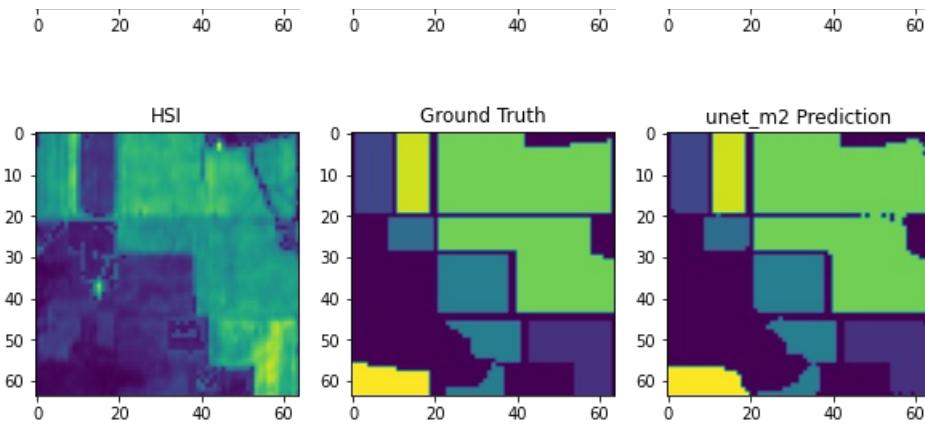
```

plt.imshow(im[:, :, 20])
plt.title('HSI')
plt.subplot(132)
plt.imshow(gt)
plt.title('Ground Truth')
plt.subplot(133)
plt.imshow(prediction[0])
plt.title('unet_m2 Prediction')
plt.colorbar(im1, ax=axis[1], shrink=0.4, aspect=16, ticks=range(0,17,1))
plt.show()
i+=1
if(i>10):
    break

```







unet_m2 prediction for complete image

Generating the segmentation of original image (145x145) from patches

In []:

```
HSI_orig_patch = img_patch_list_new[0]
HSI_orig_patch.shape
```

Out[]:

```
(10, 10, 64, 64, 95)
```

In []:

```
# Loading data associated with the original image (145x145)
HSI_orig_dataset = []
for i in range(HSI_orig_patch.shape[0]):
    for j in range(HSI_orig_patch.shape[1]):
        single_patch = HSI_orig_patch[i][j]
        single_patch = Std_scaler.transform(single_patch.reshape(-1, single_patch.shape[-1])).reshape(single_patch.shape)
        HSI_orig_dataset.append(single_patch)
```

In []:

```
# Converting original patch list to numpy array
HSI_orig_dataset = np.array(HSI_orig_dataset)
```

In []:

```
HSI_orig_dataset.shape
```

Out[]:

```
(100, 64, 64, 95)
```

In []:

```
# predicting for individual patch
pred = unet_m2.predict(HSI_orig_dataset)
prediction = np.argmax(pred, axis=-1)
```

In []:

```
pred.shape
```

Out[]:

```
(100, 64, 64, 17)
```

In []:

```
# individual patch is combined to form a grid of patches
grid = 0
img_pred = np.zeros((10, 10, 64, 64))
for i in range(10):
    for j in range(10):
        img_pred[i][j] = prediction[grid]
        grid+=1
```

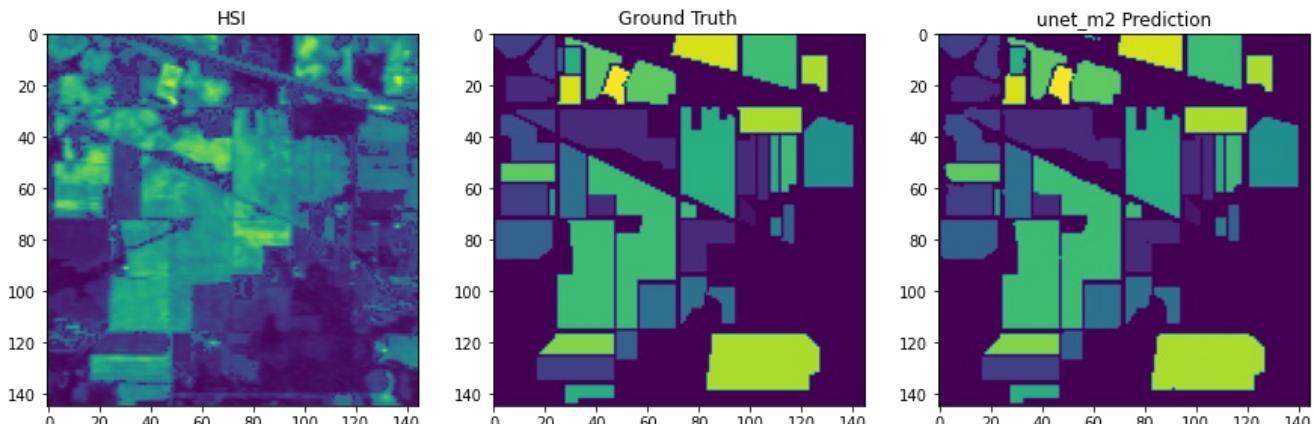
Unpatchified prediction

In []:

```
# converting the predicted patches into complete image using unpatchify
HSI_orig_pred = patch.unpatchify(img_pred, (145,145))
```

In []:

```
# plotting comparison of HSI vs Ground truth vs unet_m2 predictions
plt.figure(figsize=(15,15))
plt.subplot(131)
plt.imshow(img[:, :, 30])
plt.title('HSI')
plt.subplot(132)
plt.imshow(img_gt)
plt.title('Ground Truth')
plt.subplot(133)
plt.imshow(HSI_orig_pred)
plt.title('unet_m2 Prediction')
plt.show()
```



Note: In unpatchify method, each patch at the overlapping regions are replaced by next patch. Alternative approach for stitching all patches is presented below.

Prediction based on max score of patches

Here the segmentation is generated by constructing the matrix of size (145, 145, 100*17) where model prediction probabilities(64x64x17) of each patch are placed along third axis in a manner mentioned below:

- First patch(predictions) will be placed at (0,0,0)
- Second patch(predictions) will be placed at (0,9,17)
- Third patch(predictions) will be placed at (0,18,34) ...
- Last patch(predictions) will be placed at (137,137,1684)

This is done to consider max probability from multiple prediction for the overlapping regions. In this way the best class is selected at overlapping regions by using argmax along third axis and modulo operator for 17

In []:

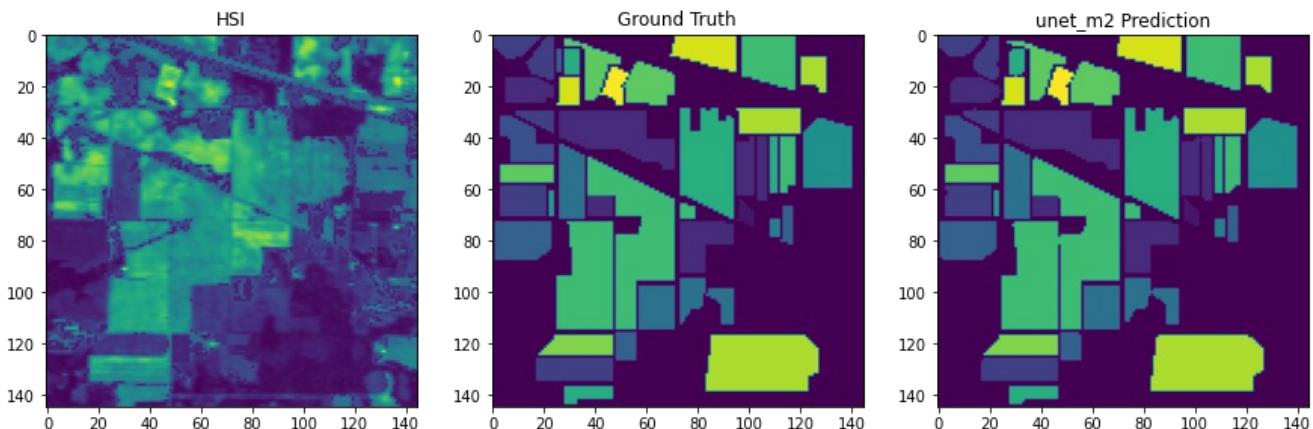
```
# Generating the 3D probabilities grid of all patches associated with full image.
grid = 0
grp = 0
img_prediction = np.zeros((145, 145, 100*17))
for i in range(10):
    for j in range(10):
        img_prediction[i*9:i*9+64,
                      j*9:j*9+64,
                      grp:grp+17] = pred[grid]
    grid+=1
    grp+=17
```

In []:

```
# Identifying the classes of each pixel from probabilities values of all patches corresponding to image
# (145x145)
prediction = np.argmax(img_prediction, axis=-1)%17
```

In []:

```
# Plotting the segmentation after identifying the best class for overlapping patches
plt.figure(figsize=(15,15))
plt.subplot(131)
plt.imshow(img[:, :, 30])
plt.title('HSI')
plt.subplot(132)
plt.imshow(img_gt)
plt.title('Ground Truth')
plt.subplot(133)
plt.imshow(prediction)
plt.title('unet_m2 Prediction')
plt.show()
```



We can observe that the segmentation is better than the unpatchify generated image. And also better than unet_m1 model

Full image prediction score (F1 and kappa)

In []:

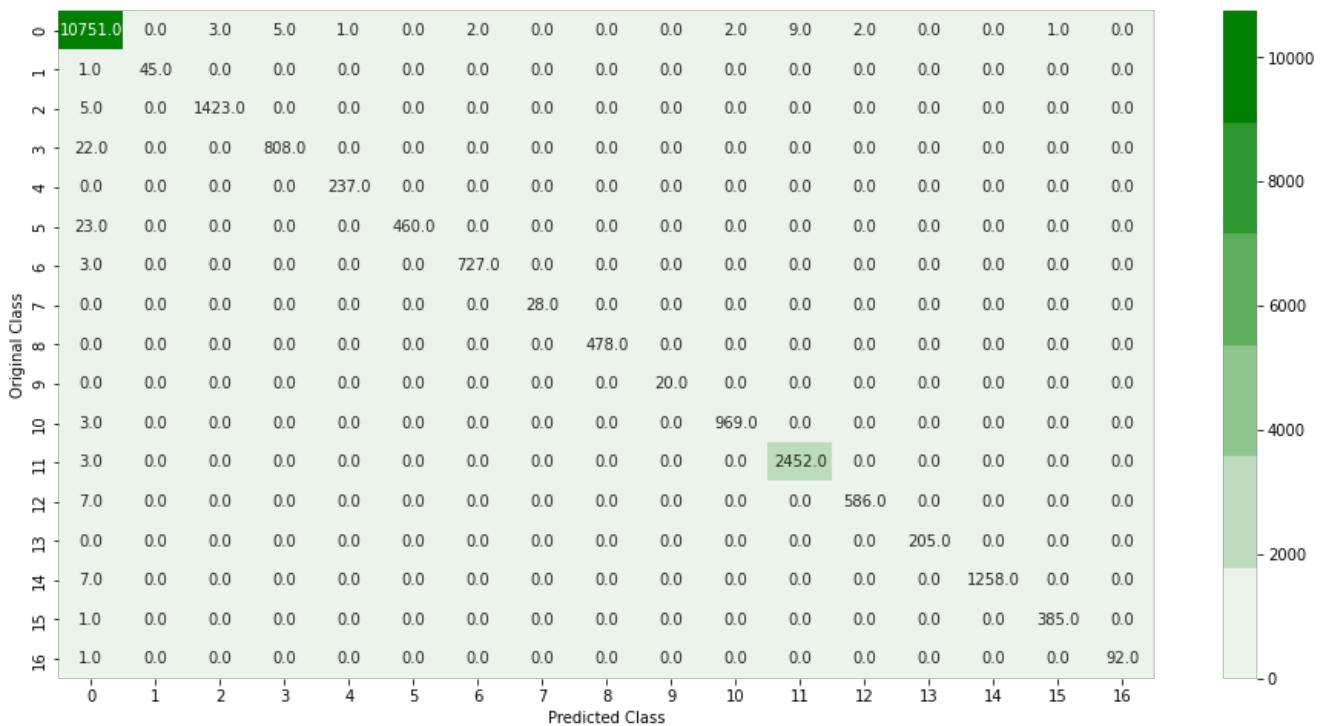
```
# Flattening the ground truths and predictions (145x145 image) for score evaluation
y = img_gt.flatten()
y_hat = prediction.flatten()
```

In []:

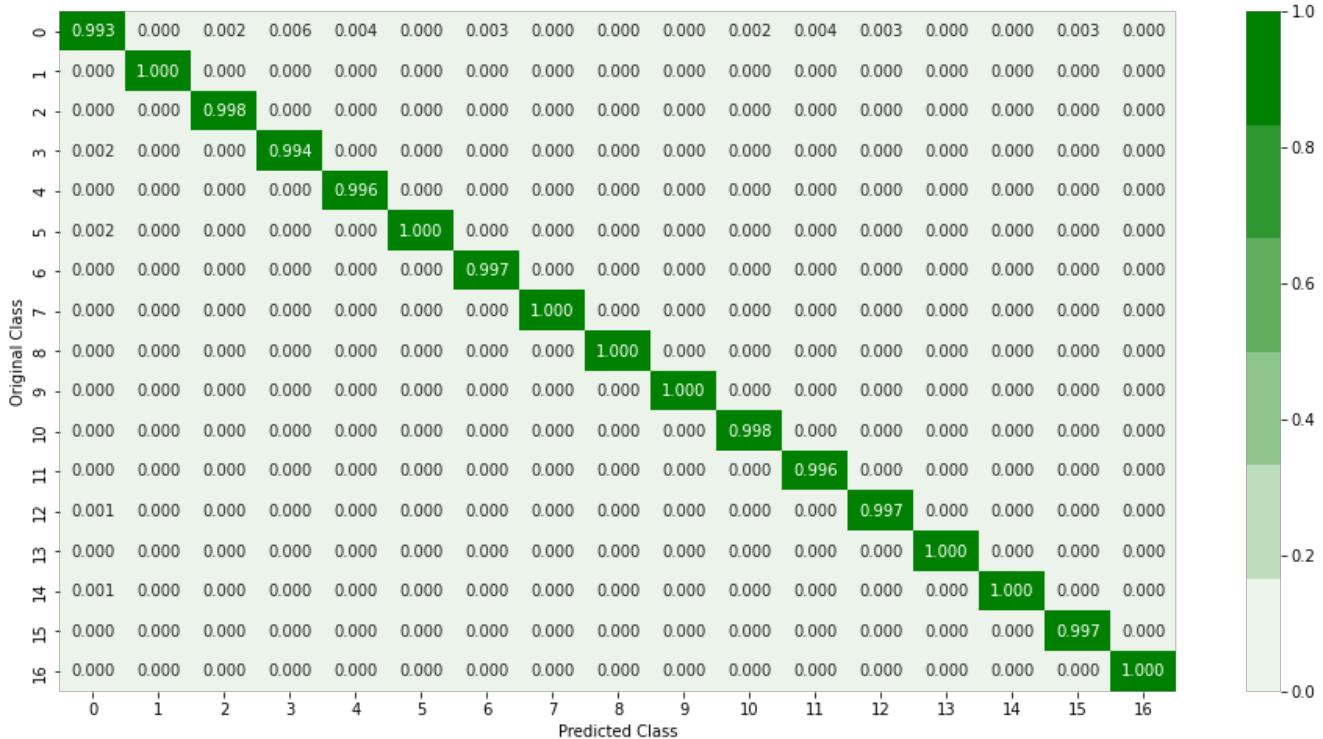
```
plot_confusion_matrix_2(y,y_hat)
```

Confusion / Precision / Recall matrix
Percentage of misclassified points 0.48038049940546973

----- Confusion matrix -----

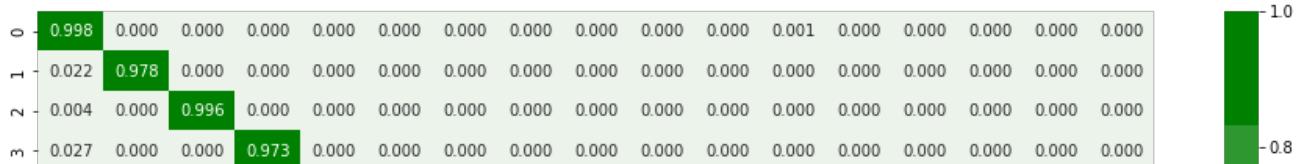


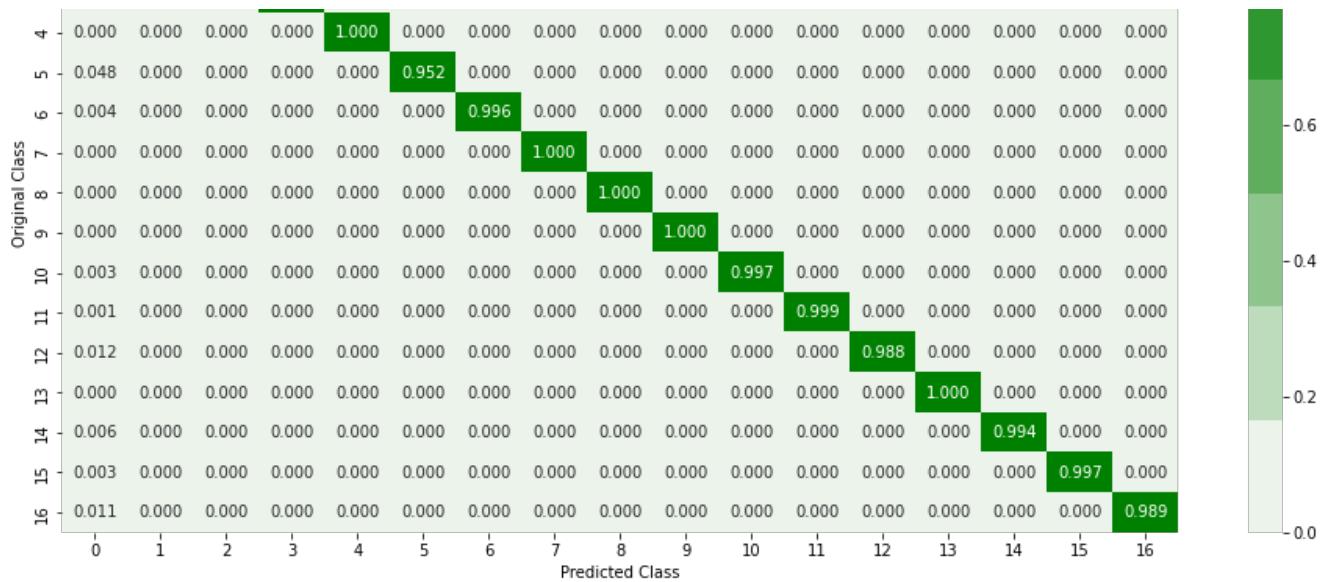
----- Precision matrix -----



Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

----- Recall matrix -----





Sum of rows in recall matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

micro F1 score : 0.9951961950059452

Average Accuracy : 0.9917185340702078

In []:

```
F1_unet_m2 = f1_score(y,y_hat,average='micro')
print('micro F1 score of simple unet model for full image : ',F1_unet_m2)
kappa_unet_m2 = cohen_kappa_score(y,y_hat)
print('kappa score of simple unet model for full image : ',kappa_unet_m2)
```

micro F1 score of simple unet model for full image : 0.9951961950059452
kappa score of simple unet model for full image : 0.9932047213699282

Validation set score

Score evaluation for the test split to understand the performance of predicting the patches

In []:

```
X_test.shape,y_test.shape
```

Out[]:

```
((200, 64, 64, 95), (200, 64, 64))
```

In []:

```
pred_test = unet_m2.predict(X_test)
prediction_test = np.argmax(pred_test,axis=-1)
```

In []:

```
prediction_test.shape
```

Out[]:

```
(200, 64, 64)
```

In []:

```
y_val = y_test.flatten()
y_hat_val = prediction_test.flatten()
```

```
y_hat_val = prediction_test.predict()
```

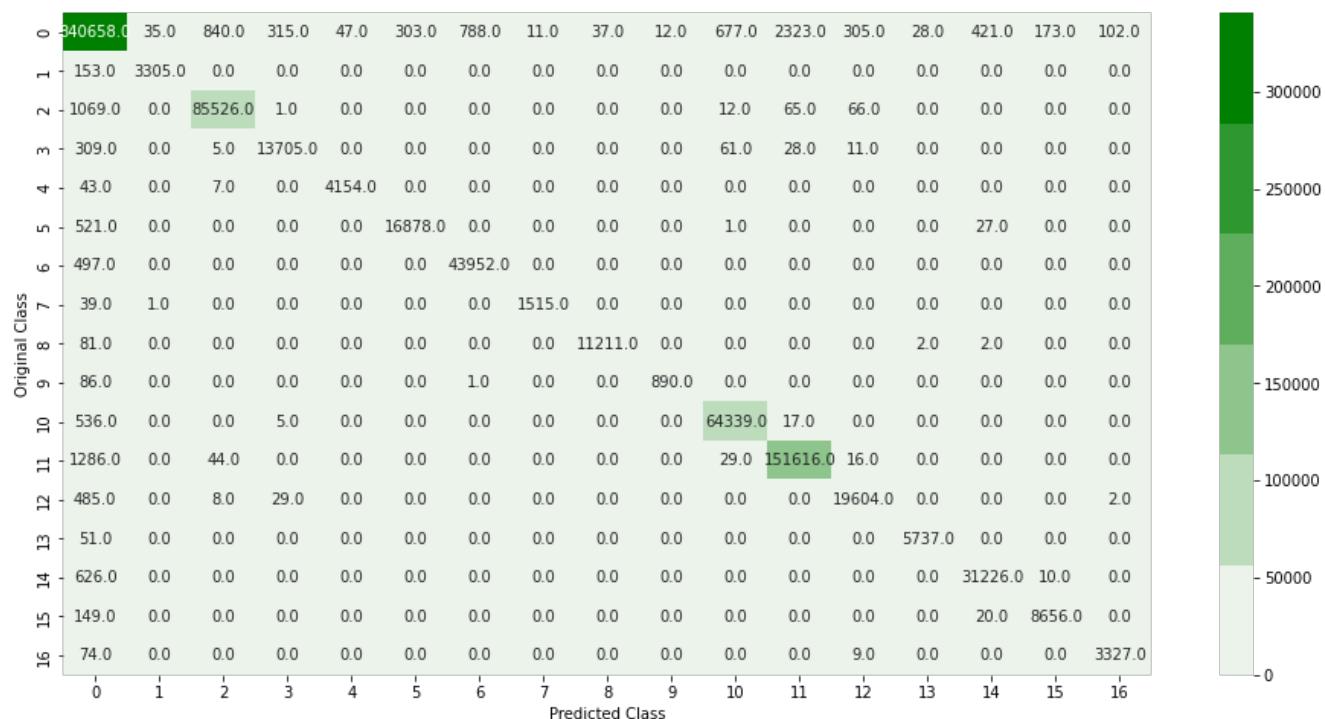
In []:

```
plot_confusion_matrix_2(y_val,y_hat_val)
```

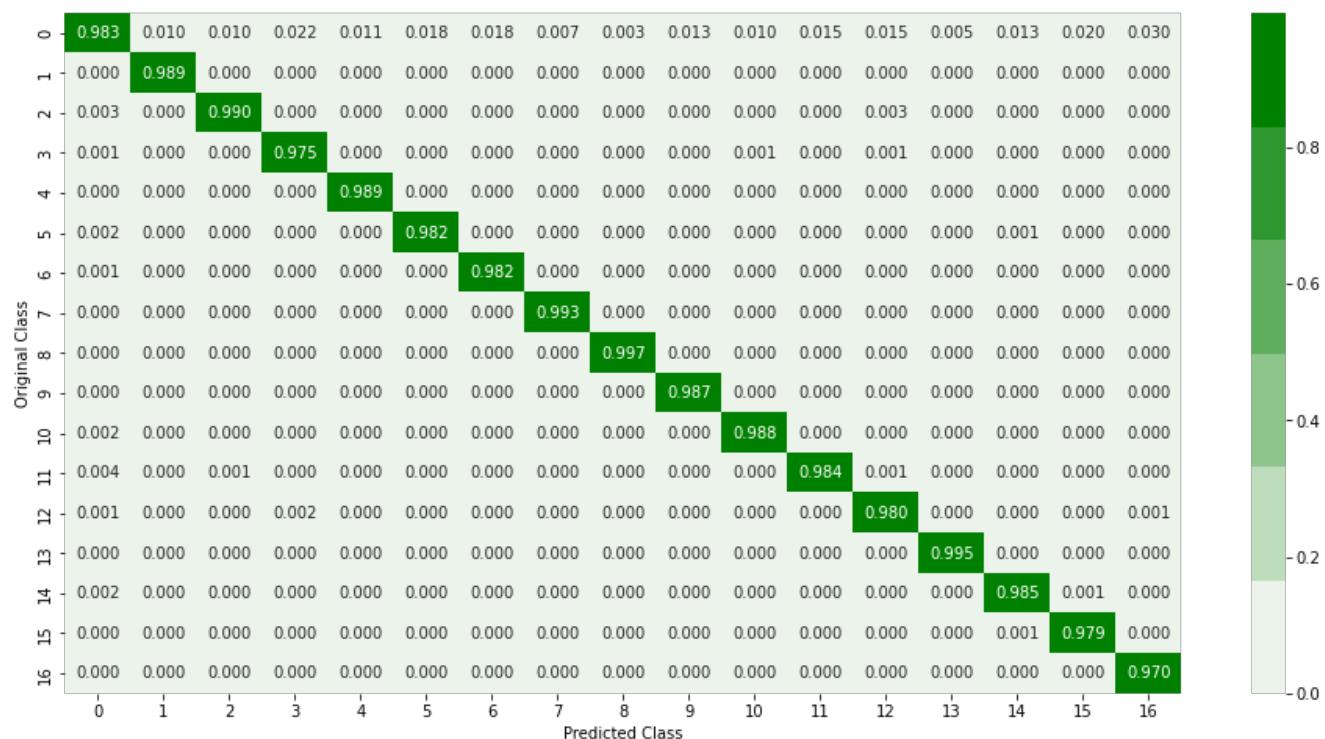
Confusion / Precision / Recall matrix

Percentage of misclassified points 1.5748291015625

----- Confusion matrix -----

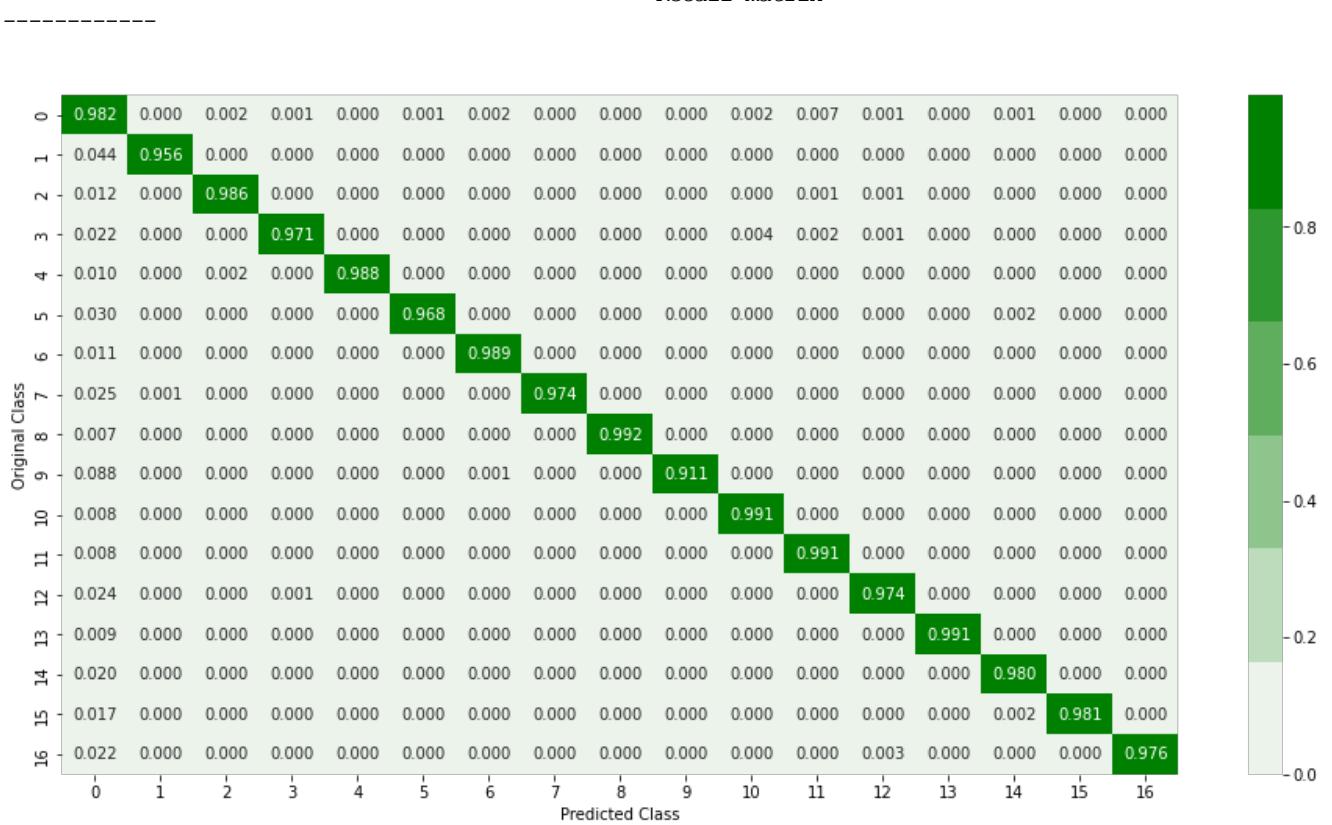


----- Precision matrix -----



Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

----- Recall matrix -----



Sum of rows in recall matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

micro F1 score : 0.984251708984375

Average Accuracy : 0.9765413816114771

In []:

```
F1_unet_m2_val = f1_score(y_val,y_hat_val,average='micro')
print('micro F1 score of simple unet model for validation data: ',F1_unet_m2_val)
kappa_unet_m2_val = cohen_kappa_score(y_val,y_hat_val)
print('kappa score of simple unet model for validation data: ',kappa_unet_m2_val)
```

micro F1 score of simple unet model for validation data: 0.984251708984375
kappa score of simple unet model for validation data: 0.9793300495581007

In []:

```
# plt.figure(figsize=(15,15))
# im_count=1
# for i in range(10):
#     for j in range(10):
#         plt.subplot(10,10,im_count)
#         plt.imshow(img_pred[i][j])
#         im_count+=1
# plt.show()
```

Testing unet_m2 model on unseen data

The score we see for the Full image segmentation is because the model has seen the class structures during the training. Its score drops for the validation set because it has some unseen data.

Point to be noted here is that the data of train and validation set comes from the same image patch with different augmentation.

The validation set will not have same image as training set but the regions of class within image will be shifted compared to the ones in train set. As the train/test split was generated from cropped images which have overlapping regions, most of the shapes of classes in the validation set are covered in train set except for few which reduced the score for validation set.

To know the true performance we need to Test the model on unseen data, where the class sizes are much different (smaller or bigger) compared to original image.

Since the only image we have here is 145×145 , we shall construct image from the 64×64 images of test set. The new image will have the test set images overlapped on each other such that a 64×64 patch will have 4 (32×32) images. This will generate a New landscape where the classes do not have shapes same as the original Indian Pines. We shall extract the 64×64 patches from this newly generated image and test the model prediction.

In []:

```
# Selecting 64 x 64 images from test set to create new 145 x 145 image
test_image = X_test[::3]
test_image_gt = y_test[::3]
test_image.shape, test_image_gt.shape
```

Out []:

```
((67, 64, 64, 95), (67, 64, 64))
```

In []:

```
# 145 x 145 image generation (augmented landscape)
grid = 0
test_image_full = np.zeros((32*6, 32*6, 95))
test_image_gt_full = np.zeros((32*6, 32*6))
for i in range(5):
    for j in range(5):
        test_image_full[i*32:i*32+64,
                      j*32:j*32+64,:] = test_image[grid]
        test_image_gt_full[i*32:i*32+64,
                           j*32:j*32+64] = test_image_gt[grid]
    grid+=1

print('Test image size before cropping',test_image_full.shape, test_image_gt_full.shape)

test_image_full = test_image_full[0:145,0:145,:]
test_image_gt_full = test_image_gt_full[0:145,0:145]
print('Test image size after cropping',test_image_full.shape, test_image_gt_full.shape)
```

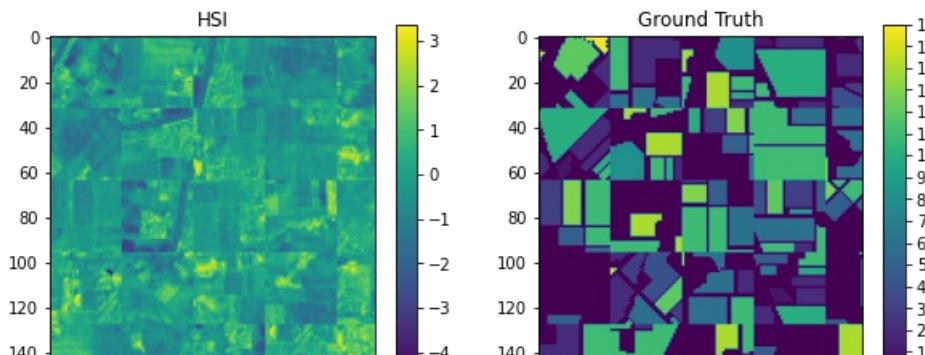
Test image size before cropping (192, 192, 95) (192, 192)
Test image size after cropping (145, 145, 95) (145, 145)

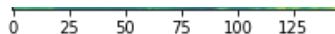
New Test Image

In []:

```
# New image
fig, axis = plt.subplots(1,2, figsize=(10,10))
im0 = axis[0].imshow(test_image_full[:, :, 30])#,cmap='jet')
axis[0].set_title('HSI')
plt.colorbar(im0, ax=axis[0], shrink=0.4, aspect=16) #, ticks=range(0,17,1))

im1 = axis[1].imshow(test_image_gt_full) #,cmap='jet')
axis[1].set_title('Ground Truth')
plt.colorbar(im1, ax=axis[1], shrink=0.4, aspect=16, ticks=range(0,17,1))
plt.show()
```





Generating patches for testing

In []:

```
# Generating the patches
test_img_pch = np.squeeze(patch.patchify(test_image_full,(64, 64,95) , step=9), axis=2)
test_img_gt_pch = patch.patchify(test_image_gt_full,(64, 64), step=9)
```

In []:

```
test_img_pch.shape,test_img_gt_pch.shape
```

Out[]:

```
((10, 10, 64, 64, 95), (10, 10, 64, 64))
```

In []:

```
# Loading data associated with the new test image (145x145)
HSI_test_dataset = []
for i in range(test_img_pch.shape[0]):
    for j in range(test_img_pch.shape[1]):
        single_patch = test_img_pch[i][j]
        # data is already standardised
        # single_patch = Std_scaler.transform(single_patch.reshape(-1,single_patch.shape[-1])).reshape(single_patch.shape)
        HSI_test_dataset.append(single_patch)
```

In []:

```
# Converting original patch list to numpy array
HSI_test_dataset = np.array(HSI_test_dataset)
```

In []:

```
# Generating Groundtruth dataset seperating the single 64x64 patch from patch grid (10,10,64,64)
HSI_test_gt_dataset = []
for i in range(test_img_gt_pch.shape[0]):
    for j in range(test_img_gt_pch.shape[1]):
        HSI_test_gt_dataset.append(patchs[i][j])
```

In []:

```
# Converting original gt patch list to numpy array
HSI_test_gt_dataset = np.array(HSI_test_gt_dataset)
```

Model Prediction for the new test image patches

In []:

```
%%timeit
# predicting for individual patch
pred = unet_m2.predict(HSI_test_dataset)
```

```
1 loop, best of 5: 406 ms per loop
```

In []:

```
pred = unet_m2.predict(HSI_test_dataset)
```

```
In [ ]:  
pred.shape
```

```
Out[ ]:  
(100, 64, 64, 17)
```

Reconstructing the 145 x 145 image predictions

```
In [ ]:
```

```
# Generating the 3D probabilities grid of all patches associated with full image.  
grid = 0  
grp = 0  
img_prediction = np.zeros((145, 145, 100*17))  
for i in range(10):  
    for j in range(10):  
        img_prediction[i*9:i*9+64,  
                      j*9:j*9+64,  
                      grp:grp+17] = pred[grid]  
    grid+=1  
    grp+=17  
  
img_prediction.shape
```

```
Out[ ]:  
(145, 145, 1700)
```

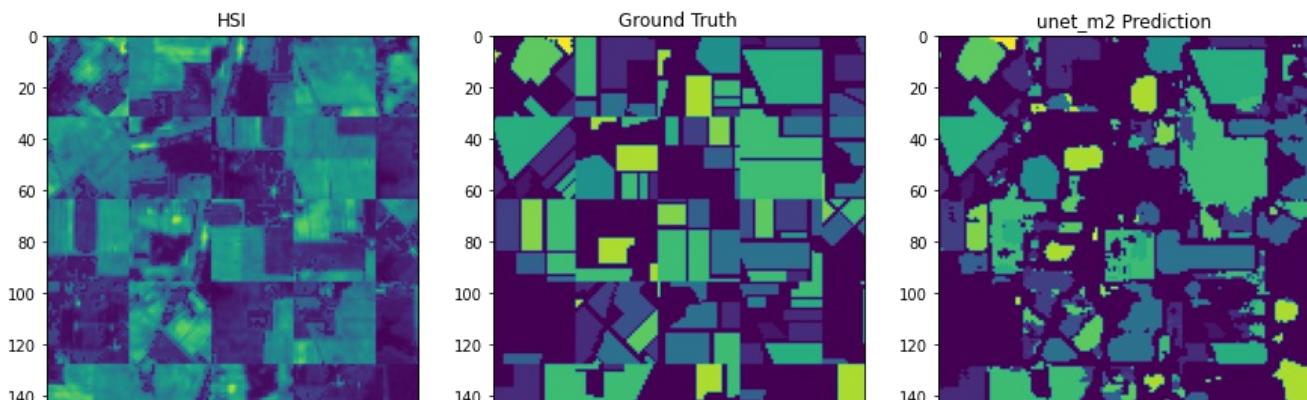
```
In [ ]:
```

```
# Identifying the classes of each pixel from probabilities values of all patches corresponding to image  
(145x145)  
prediction = np.argmax(img_prediction, axis=-1)%17
```

Prediction

```
In [ ]:
```

```
# Plotting the segmentation after identifying the best class for overlapping patches  
plt.figure(figsize=(15,15))  
plt.subplot(131)  
plt.imshow(test_image_full[:, :, 20])  
plt.title('HSI')  
plt.subplot(132)  
plt.imshow(test_image_gt_full)  
plt.title('Ground Truth')  
plt.subplot(133)  
plt.imshow(prediction)  
plt.title('unet_m2 Prediction')  
plt.show()
```





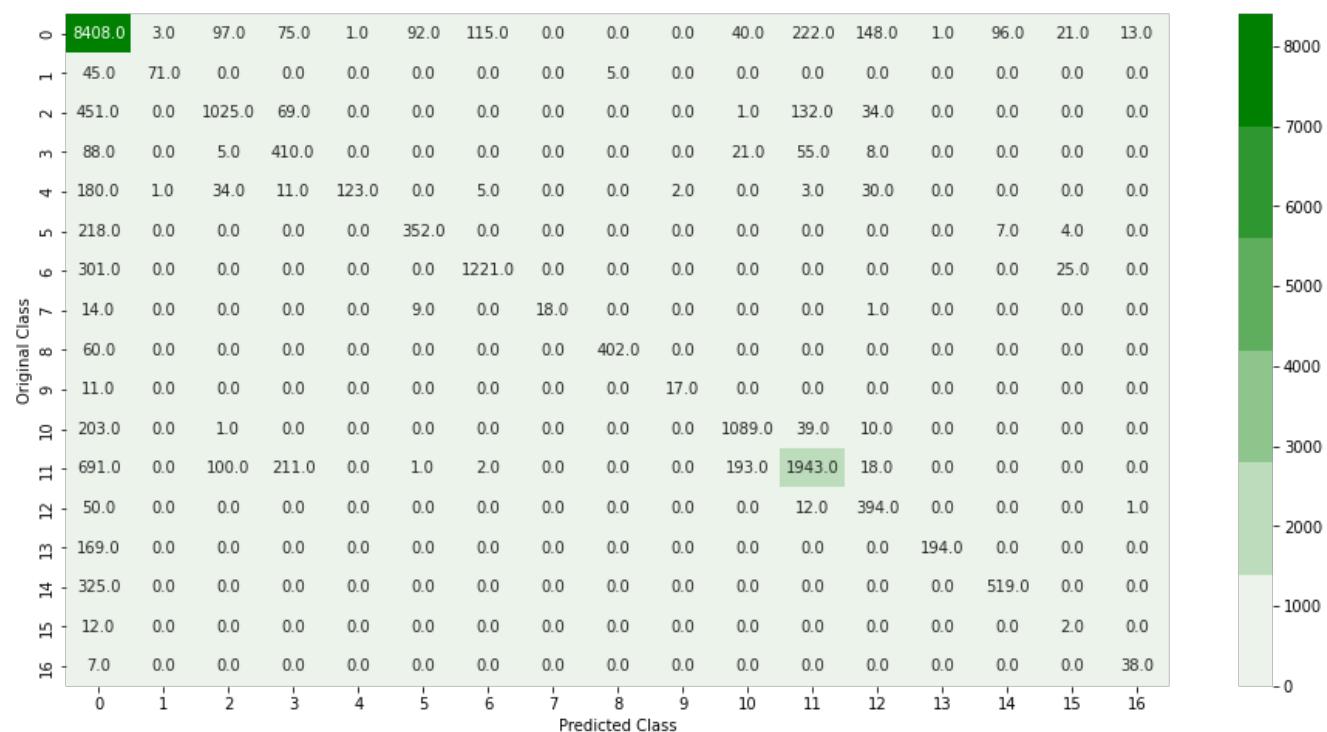
Modified image prediction score (F1 and kappa)

In []:

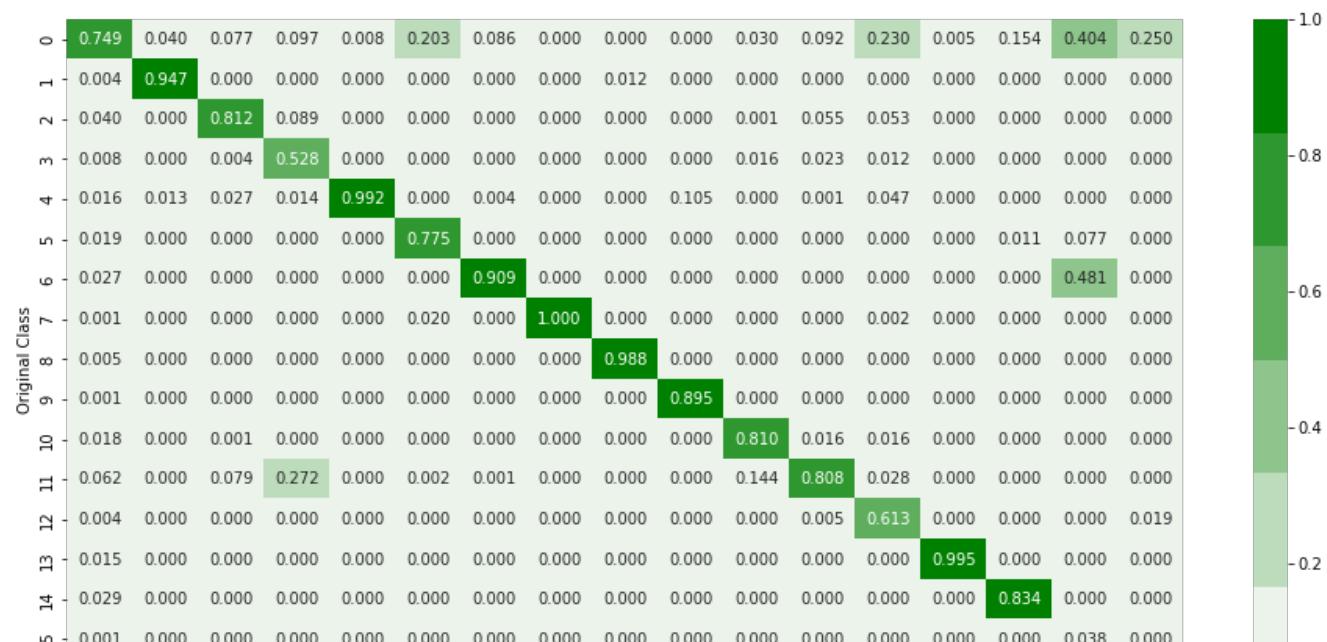
```
# Flattening the ground truths and predictions (145x145 image) for score evaluation
y = test_image_gt_full.flatten()
y_hat = prediction.flatten()
plot confusion matrix 2(y,y_hat)
```

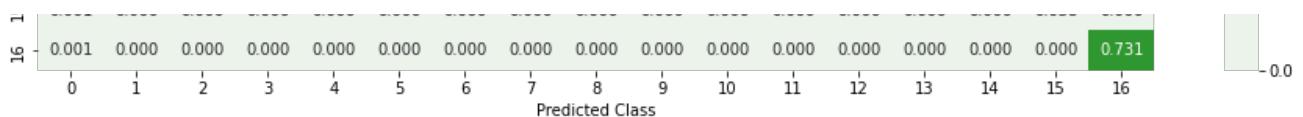
Confusion / Precision / Recall matrix
Percentage of misclassified points 22.825208085612367

- Confusion matrix



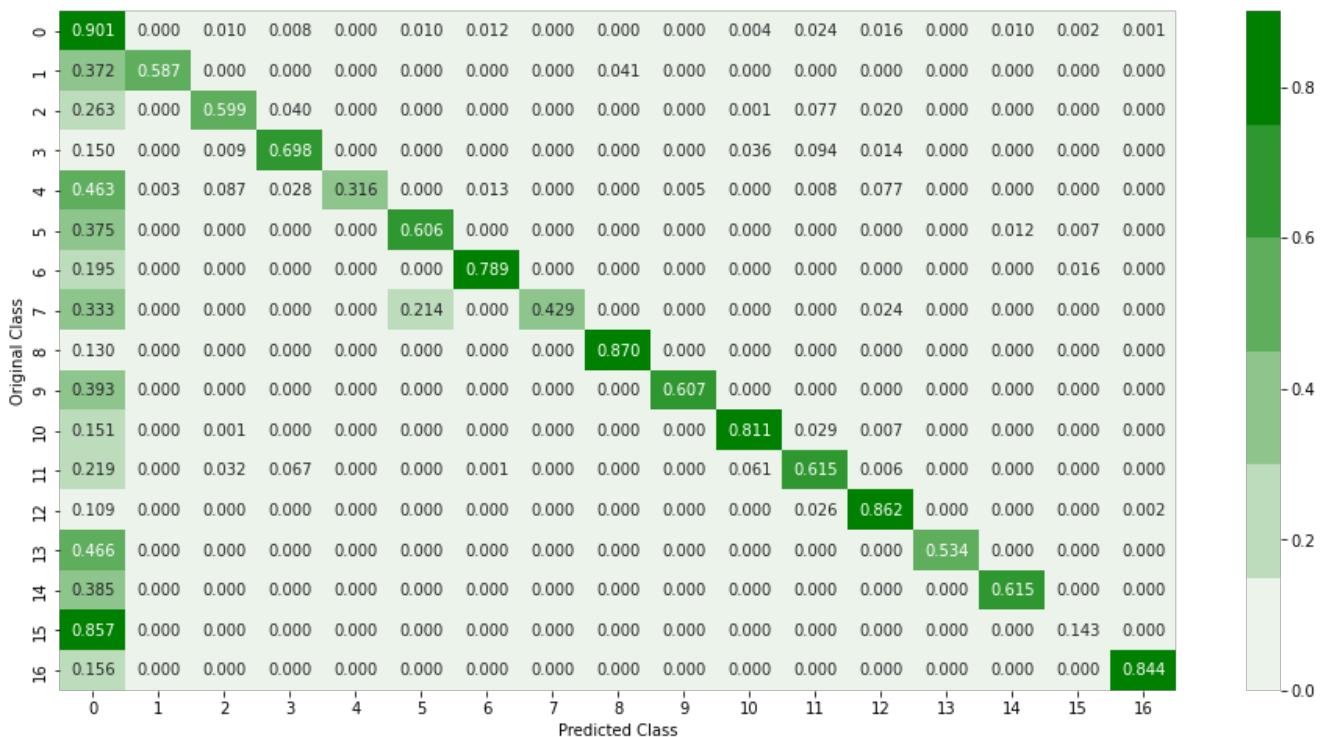
- Precision matrix





Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

----- Recall matrix -----



Sum of rows in recall matrix [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

micro F1 score : 0.7717479191438763

Average Accuracy : 0.6369093622643562

Model is unable to identify and segment the class 15. Also Most of other classes are categorised as class 0 by the model.

In []:

```
F1_unet_m2 = f1_score(y,y_hat,average='micro')
print('micro F1 score of simple unet model for test image : ',F1_unet_m2)
kappa_unet_m2 = cohen_kappa_score(y,y_hat)
print('kappa score of simple unet model for test image : ',kappa_unet_m2)
```

micro F1 score of simple unet model for test image : 0.7717479191438763
kappa score of simple unet model for test image : 0.6863796574566523

Observations:

1. Pretrained U-Net

Model was trained for 50 epochs

- Scores for Full image prediction (Train and Validation data combined):
 - micro F1 score : 87.22%
 - Average Accuracy : 72.22%
 - kappa score : 81.56%
- Scores for test set image prediction (only validation data):
 - micro F1 score : 0.7717479191438763
 - Average Accuracy : 0.6369093622643562
 - kappa score : 0.6863796574566523

- micro F1 score : 82.30%
 - Average Accuracy : 64.89%
 - kappa score : 76.34%
 - Scores for new augmented image prediction :
 - micro F1 score : 64.52%
 - Average Accuracy : 30.07%
 - kappa score : 48.55%
 - Though the scores are better for full image, the segmented images are more like globules. This might be due to the model which have trained weights(imagenet) that are trained specifically for RGB images. While the 3 channel image input in current problem is reduced from 95 channel image.
 - The performance reduces further for unseen augmented landscape image
-

1. Simple Unet trained from scratch

Model was trained for 50 epochs and retrained for additional 50 epochs to get better result.

- Scores for Full image prediction (Train and Validation data combined):
 - micro F1 score : 99.51%
 - Average Accuracy : 99.17%
 - kappa score : 99.32%
 - Scores for test image prediction (only validation data):
 - micro F1 score : 98.42%
 - Average Accuracy : 97.65%
 - kappa score : 97.93%
 - Scores for new augmented image prediction :
 - micro F1 score : 77.17%
 - Average Accuracy : 63.69%
 - kappa score : 68.63%
 - This model which is trained from scratch are able to segment the HS image very well. The predicted image are pretty indistinguishable from ground truth. This model can only be used to classify the Hyperspectral Images which have mentioned 16 classes of Indian Pines. Input for the model must be 64x64x95.
 - The performance reduced for unseen augmented landscape image, since the shapes of classes(gt) are not similar to the train set.
-

For Classifying the HS Image of broader classes, Simple U-Net model can be considered and trained from scratch for a larger dataset.

These models will be dedicated for Hyper Spectral Image segmentation of specific class set.