Hyper Spectral Image Segmentation using UNET

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 [ ]:

```
pip install patchify
```

```
Collecting patchify
  Downloading patchify-0.2.3-py3-none-any.whl (6.6 kB)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.7/dist-packages (from patchify) (1
.21.5)
Installing collected packages: patchify
Successfully installed patchify-0.2.3
```

In [ ]:

```
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 [ ]:

```
# 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,a
pplication/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signe
d-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/Indi
an_pines_corrected.mat" -c -O 'Indian_pines_corrected.mat'
!unzip Indian_pines_corrected.mat
```

```
--2022-03-04 05:17:56--  http://wget/
Resolving wget (wget)... failed: Name or service not known.
wget: unable to resolve host address ‘wget’
```

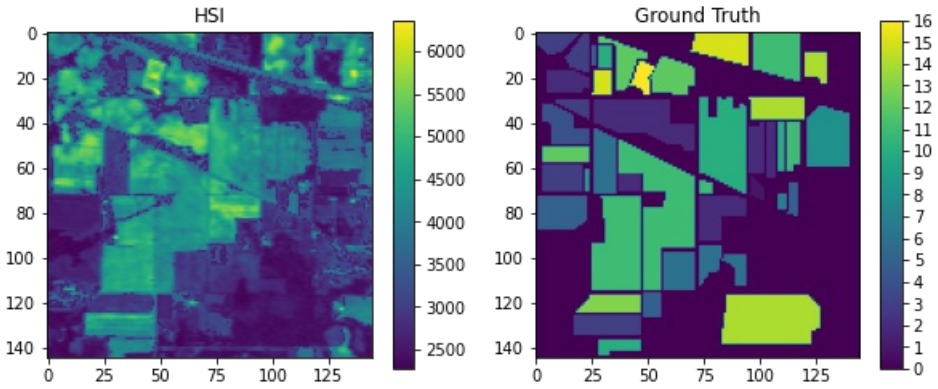
```
--2022-03-04 05:17:56--  http://www.ehu.eus/ccwintco/uploads/6/67/Indian_pines_corrected.mat
Resolving www.ehu.eus (www.ehu.eus)... 158.227.0.65, 2001:720:1410::65
Connecting to www.ehu.eus (www.ehu.eus)|158.227.0.65|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5953527 (5.7M)
Saving to: 'Indian_pines_corrected.mat'

Indian_pines_correc 100%[===================>]   5.68M   752KB/s    in 8.6s

2022-03-04 05:18:05 (673 KB/s) - 'Indian_pines_corrected.mat' saved [5953527/5953527]

FINISHED --2022-03-04 05:18:05--
Total wall clock time: 9.2s
Downloaded: 1 files, 5.7M in 8.6s (673 KB/s)
Archive:  Indian_pines_corrected.mat
  End-of-central-directory signature not found.  Either this file is not
  a zipfile, or it constitutes one disk of a multi-part archive.  In the
  latter case the central directory and zipfile comment will be found on
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of Indian_pines_corrected.mat or
        Indian_pines_corrected.mat.zip, and cannot find Indian_pines_corrected.mat.ZIP, period.
```

In [ ]:

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

In [ ]:

```
figr,axis = plt.subplots(1,2,figsize=(10,10))
im0 = axis[0].imshow(img[:,:,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)#,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')
img_gt_rot4 =  rotate(img_gt,-45,reshape=False,mode ='reflect')
img_gt_rot4.shape
```

Out[ ]:

```
(145, 145)
```

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 croped 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,          (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[ ]:

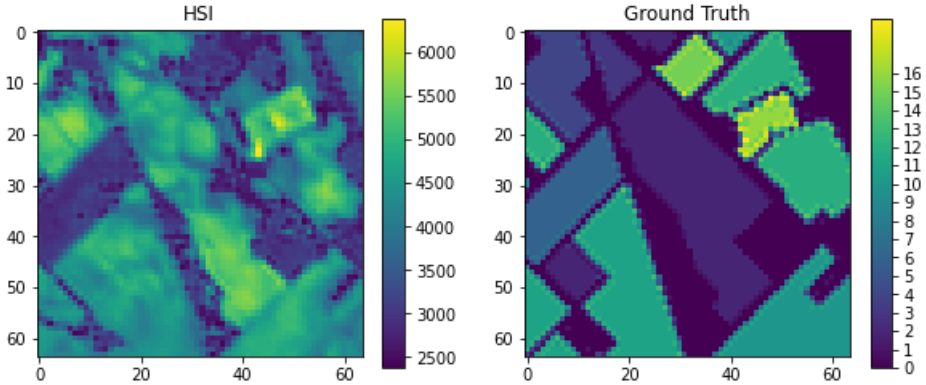
```
(64, 64)
```

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[0][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[0][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

In [ ]:

```
!wget --header="Host: doc-08-9k-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" --heade
r="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="Cooki
e: AUTH 82jcsesreiehjbhkrct3c4mrj1raokod nonce=q8qh1ahtf4f6a" --header="Connection: keep-alive" "https:
```

```
//doc-08-9k-docs.googleusercontent.com/docs/securesc/rg90kivf62vcrm9d2s7vb24hsj0c3fo2/l07odngss28mjnkteaojfea7h3m8qjtv/1646371125000/00176583124175523585/00176583124175523585/1x4XcTWHS3r7XAJgDRYaeYqk6xs_MAzut?e=download&ax=ACxEAsYzzyM7hHiUKXw0srY_rRqK7tk9Q9dScX24kqpjjUIzC02aIf-qn_qrOjGrCkGqP2w7b2ALJiFs339Ng4_wVB4gsrnxjmXED7dGfpx4kLIISogAwiYTcLyR4L5lanaansoSCWhmpX-fMu-FKbllv3XmyWODw09YSRauDh5BduTD23Ntoj458saACTXhDoK0BFfdLwVY8skWQvqdpl-pqmL0xeXuvUingN_XwEmlplEjdm12jLEZwC2i66CTEAlynV9MQ3-PPNWAY53w9zbYTvthGHQDwE19hjvAwc4i1D4nsGBxjZzYMMNOPopYbriZ3DWHNe64hEDqO8fNioUwxN9jfyKYylszgHCAhgh0PnDxkH2b1gjp69CmunBLKPAQhD5mojKGsXdvbzGtisM4S6xapHDo1G0HaT8SHhkbVdZxw6zIu0VWkORu1JE-AFrzigsi00qbigTEuqreDpBfs-HE-TIYsLC70yPUCkkwu5XYuITb4XHlG3U_agU4oWAUkCMBThmKi1O21UTwkMbv6dHR6zxh35z8ajWcKZtpdL7QCk037tqDBKhYAMtMHcz33mU1ZTX-ZLL2fr7zvPxgrcKdQs5Cj_ehijr4RARZ10WKVIzR_NhcJN8D7281B2RnOtnNz3R9Fmmna58pn5quOD0VIouSTZmrmYNHbSBZ&authuser=0&nonce=q8gh1ahtf4f6a&user=00176583124175523585&hash=08tleqn6p734fp3mo0ird8m361qnvetv" -c -O 'Indian_pines_HSI_AUGM_1to10.zip'
!unzip Indian_pines_HSI_AUGM_1to10.zip
```

```
--2022-03-04 05:19:10--  https://doc-08-9k-docs.googleusercontent.com/docs/securesc/rg90kivf62vcrm9d2s7vb24hsj0c3fo2/l07odngss28mjnkteaojfea7h3m8qjtv/1646371125000/00176583124175523585/00176583124175523585/1x4XcTWHS3r7XAJgDRYaeYqk6xs_MAzut?e=download&ax=ACxEAsYzzyM7hHiUKXw0srY_rRqK7tk9Q9dScX24kqpjjUIzC02aIf-qn_qrOjGrCkGqP2w7b2ALJiFs339Ng4_wVB4gsrnxjmXED7dGfpx4kLIISogAwiYTcLyR4L5lanaansoSCWhmpX-fMu-FKbllv3XmyWODw09YSRauDh5BduTD23Ntoj458saACTXhDoK0BFfdLwVY8skWQvqdpl-pqmL0xeXuvUingN_XwEmlplEjdm12jLEZwC2i66CTEAlynV9MQ3-PPNWAY53w9zbYTvthGHQDwE19hjvAwc4i1D4nsGBxjZzYMMNOPopYbriZ3DWHNe64hEDqO8fNioUwxN9jfyKYylszgHCAhgh0PnDxkH2b1gjp69CmunBLKPAQhD5mojKGsXdvbzGtisM4S6xapHDo1G0HaT8SHhkbVdZxw6zIu0VWkORu1JE-AFrzigsi00qbigTEuqreDpBfs-HE-TIYsLC70yPUCkkwu5XYuITb4XHlG3U_agU4oWAUkCMBThmKi1O21UTwkMbv6dHR6zxh35z8ajWcKZtpdL7QCk037tqDBKhYAMtMHcz33mU1ZTX-ZLL2fr7zvPxgrcKdQs5Cj_ehijr4RARZ10WKVIzR_NhcJN8D7281B2RnOtnNz3R9Fmmna58pn5quOD0VIouSTZmrmYNHbSBZ&authuser=0&nonce=q8gh1ahtf4f6a&user=00176583124175523585&hash=08tleqn6p734fp3mo0ird8m361qnvetv
Resolving doc-08-9k-docs.googleusercontent.com (doc-08-9k-docs.googleusercontent.com)... 142.251.6.132, 2607:f8b0:4001:c5a::84
Connecting to doc-08-9k-docs.googleusercontent.com (doc-08-9k-docs.googleusercontent.com)|142.251.6.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 412730136 (394M) [application/x-zip-compressed]
Saving to: 'Indian_pines_HSI_AUGM_1to10.zip'

Indian_pines_HSI_AU 100%[===================>] 393.61M   124MB/s    in 3.3s

2022-03-04 05:19:14 (120 MB/s) - 'Indian_pines_HSI_AUGM_1to10.zip' saved [412730136/412730136]

Archive:  Indian_pines_HSI_AUGM_1to10.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
```

Loading the data from *.mat files

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

In [ ]:

```
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 [ ]:

```
# 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 [ ]:

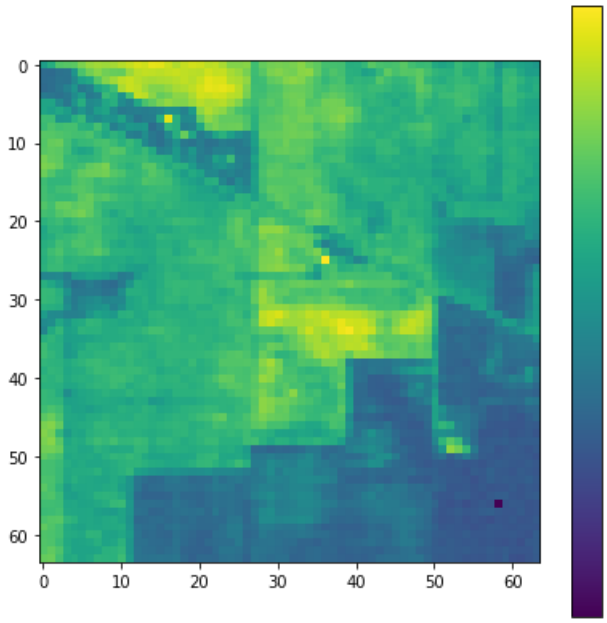
```
HSI_AUGM_1.shape
```

Out[ ]:

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

In [ ]:

```
# Example plot
plt.figure(figsize=(7,7))
plt.imshow(HSI_AUGM_1[5][5][:,:,10])
plt.colorbar(ticks=range(0,17))
plt.show()
```



In [ ]:

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

In [ ]:

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

Out[ ]:

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

In [ ]:

In [ ]:

```
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 [ ]:

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

Out[ ]:

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

In [ ]:

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

Out[ ]:

```
(21025, 200)
```

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

In [ ]:

```
# 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, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190]
```

In [ ]:

```
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 [ ]:

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

Out[ ]:

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

In [ ]:

```
# 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

Standardization

Standaradizing the values of the image matrix for each band

In [ ]:

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

In [ ]:

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

Out[ ]:

```
StandardScaler()
```

Creating Dataset to have collection of images instead of patches

In [ ]:

```
# 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(sing
le_patch.shape)
      image_dataset.append(single_patch)
```

In [ ]:

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

Out[ ]:

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

In [ ]:

```
# 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 [ ]:

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

Out[ ]:

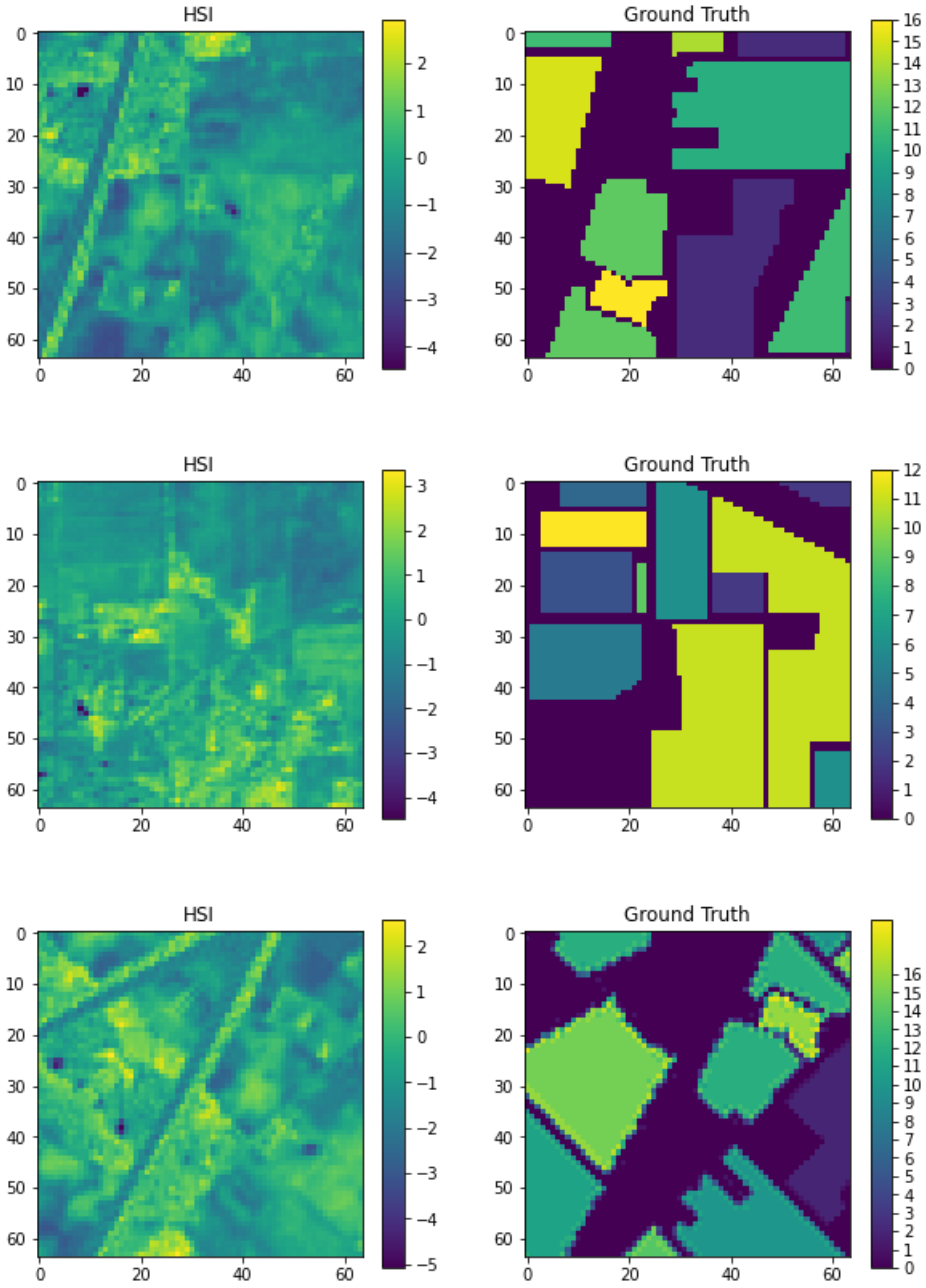
```
(1000, 64, 64)
```

Dataset Review

In [ ]:

```
for i in [150,550,900]:
```

```
for i in [150,330,900]:
  figr,axis = plt.subplots(1,2,figsize=(10,10))
  im0 = axis[0].imshow(image_dataset[i][:,:,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])#,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 [ ]:

```
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 [ ]:

```
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 [ ]:

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

In [ ]:

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

In [ ]:

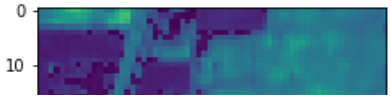
```
ex[1][:,:,10].any()
```

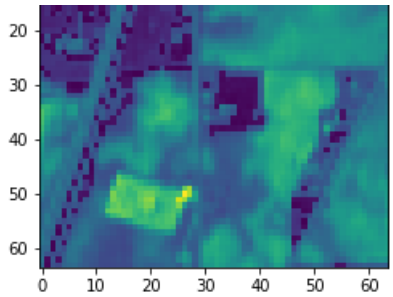
Out[ ]:

```
True
```

In [ ]:

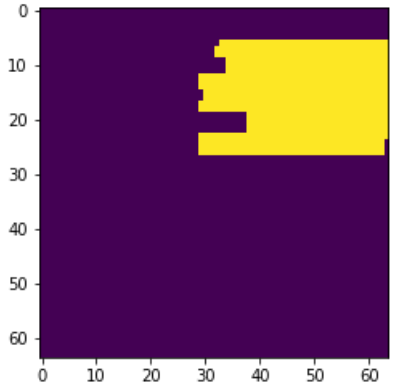
```
plt.imshow(ex[0][:,:,15])
plt.show()
```





In [ ]:

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



In [ ]:

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

In [ ]:

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

In [ ]:

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

Out[ ]:

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

Train and Test split of data

Data are split into 80% train and 20% test

In [ ]:

```
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 [ ]:

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

Out[ ]:

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

Dataset generation

In [ ]:

```
# Dataset for train images
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)
```

Unet Models

Model 1 - Pretrained model

Here pretrained model is defined using segmentation_models module

Model Definition

In [ ]:

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

```
Collecting segmentation-models
  Downloading segmentation_models-1.0.1-py3-none-any.whl (33 kB)
Collecting efficientnet==1.0.0
  Downloading efficientnet-1.0.0-py3-none-any.whl (17 kB)
Collecting image-classifiers==1.0.0
  Downloading image_classifiers-1.0.0-py3-none-any.whl (19 kB)
Collecting keras-applications<=1.0.8,>=1.0.7
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 5.7 MB/s
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages (from efficientne
t==1.0.0->segmentation-models) (0.18.3)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages (from keras-appli
cations<=1.0.8,>=1.0.7->segmentation-models) (1.21.5)
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: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py->ke
ras-applications<=1.0.8,>=1.0.7->segmentation-models) (1.5.2)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from scikit
-image->efficientnet==1.0.0->segmentation-models) (1.2.0)
Requirement already satisfied: scipy>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from scikit-imag
e->efficientnet==1.0.0->segmentation-models) (1.4.1)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (fro
m scikit-image->efficientnet==1.0.0->segmentation-models) (3.2.2)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/dist-packages (from scik
it-image->efficientnet==1.0.0->segmentation-models) (2021.11.2)
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: networkx>=2.0 in /usr/local/lib/python3.7/dist-packages (from scikit-ima
ge->efficientnet==1.0.0->segmentation-models) (2.6.3)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-im
age->efficientnet==1.0.0->segmentation-models) (2.4.1)
```

```
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from mat
plotlib!=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 matplo
tlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (1.3.2)
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: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dis
t-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (3.
0.7)
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)
Installing collected packages: keras-applications, image-classifiers, efficientnet, segmentation-models
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras-applications-1.0.8 segmentation
-models-1.0.1
```

In [ ]:

```
# 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 Unet
from tensorflow.keras.layers import Input,Conv2D,MaxPooling2D,Conv2DTranspose,concatenate,Cropping2D,Ze
roPadding2D
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.
```

In [ ]:

```
# del unet_m1
```

In [ ]:

```
# loading the unet model and using the resnet 34 and initilized weights with imagenet weights
# "classes" :different types of classes in the dataset

base_model = Unet('resnet34', encoder_weights='imagenet', classes=17, activation='softmax', input_shape
=(64,64,3),
             encoder_freeze = True)

inp = Input(shape=(64, 64, 95))

l1 = Conv2D(64, (1, 1))(inp)
l1 = Conv2D(32, (1, 1))(l1)
l1 = Conv2D(16, (1, 1))(l1)
l1 = Conv2D(8, (1, 1))(l1)
l1 = Conv2D(3, (1, 1))(l1)# map N channels data to 3 channels

out = base_model(l1)

unet_m1 = Model(inp, out, name=base_model.name)

unet_m1.summary()
```

```
Downloading data from https://github.com/qubvel/classification_models/releases/download/0.0.1/resnet34_
imagenet_1000_no_top.h5
85524480/85521592 [==============================] - 1s 0us/step
85532672/85521592 [==============================] - 1s 0us/step
Model: "model_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_1 (InputLayer)         [(None, 64, 64, 95)]      0

 conv2d (Conv2D)              (None, 64, 64, 64)        6144
```

```
 conv2d_1 (Conv2D)            (None, 64, 64, 32)        2080

 conv2d_2 (Conv2D)            (None, 64, 64, 16)        528

 conv2d_3 (Conv2D)            (None, 64, 64, 8)         136

 conv2d_4 (Conv2D)            (None, 64, 64, 3)         27

 model_1 (Functional)         (None, 64, 64, 17)        24458474

=================================================================
Total params: 24,467,389
Trainable params: 3,178,295
Non-trainable params: 21,289,094
_________________________________________________________________
```

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_m1.compile(optim, focal_loss, metrics=[iou_score])
```

Model Training

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_m1 = ModelCheckpoint('model_1_save/unet_m1_best_model_e{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,verbo
se=1)

callbacks_m1 = [checkpoint_m1, Reduce_LR_m1, tensorboard]

start = time.time()
history_m1 = unet_m1.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)
```

```
20220303-103055
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.
```

```
Epoch 1/50
80/80 [==============================] - ETA: 0s - loss: 1.1136 - iou_score: 0.0236
Epoch 1: saving model to model_1_save/unet_m1_best_model_e01.h5
80/80 [==============================] - 35s 204ms/step - loss: 1.1136 - iou_score: 0.0236 - val_loss:
1.1177 - val_iou_score: 0.0213 - lr: 1.0000e-04
Epoch 2/50
80/80 [==============================] - ETA: 0s - loss: 1.0339 - iou_score: 0.0490
```

```
Epoch 2: saving model to model_1_save/unet_m1_best_model_e02.h5
80/80 [==============================] - 16s 197ms/step - loss: 1.0339 - iou_score: 0.0490 - val_loss:
1.1456 - val_iou_score: 0.0205 - lr: 1.0000e-04
Epoch 3/50
80/80 [==============================] - ETA: 0s - loss: 0.9616 - iou_score: 0.0830
Epoch 3: saving model to model_1_save/unet_m1_best_model_e03.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.9616 - iou_score: 0.0830 - val_loss:
1.1566 - val_iou_score: 0.0238 - lr: 1.0000e-04
Epoch 4/50
80/80 [==============================] - ETA: 0s - loss: 0.8957 - iou_score: 0.1207
Epoch 4: saving model to model_1_save/unet_m1_best_model_e04.h5
80/80 [==============================] - 15s 190ms/step - loss: 0.8957 - iou_score: 0.1207 - val_loss:
1.1185 - val_iou_score: 0.0291 - lr: 1.0000e-04
Epoch 5/50
80/80 [==============================] - ETA: 0s - loss: 0.8349 - iou_score: 0.1621
Epoch 5: saving model to model_1_save/unet_m1_best_model_e05.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.8349 - iou_score: 0.1621 - val_loss:
1.0825 - val_iou_score: 0.0385 - lr: 1.0000e-04
Epoch 6/50
80/80 [==============================] - ETA: 0s - loss: 0.7927 - iou_score: 0.1943
Epoch 6: saving model to model_1_save/unet_m1_best_model_e06.h5
80/80 [==============================] - 15s 187ms/step - loss: 0.7927 - iou_score: 0.1943 - val_loss:
1.0519 - val_iou_score: 0.0510 - lr: 1.0000e-04
Epoch 7/50
80/80 [==============================] - ETA: 0s - loss: 0.7591 - iou_score: 0.2207
Epoch 7: saving model to model_1_save/unet_m1_best_model_e07.h5

Epoch 7: ReduceLROnPlateau reducing learning rate to 8.999999772640876e-05.
80/80 [==============================] - 15s 188ms/step - loss: 0.7591 - iou_score: 0.2207 - val_loss:
0.9654 - val_iou_score: 0.0854 - lr: 1.0000e-04
Epoch 8/50
80/80 [==============================] - ETA: 0s - loss: 0.7253 - iou_score: 0.2470
Epoch 8: saving model to model_1_save/unet_m1_best_model_e08.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.7253 - iou_score: 0.2470 - val_loss:
0.9109 - val_iou_score: 0.1169 - lr: 9.0000e-05
Epoch 9/50
80/80 [==============================] - ETA: 0s - loss: 0.6923 - iou_score: 0.2736
Epoch 9: saving model to model_1_save/unet_m1_best_model_e09.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.6923 - iou_score: 0.2736 - val_loss:
0.8368 - val_iou_score: 0.1621 - lr: 9.0000e-05
Epoch 10/50
80/80 [==============================] - ETA: 0s - loss: 0.6632 - iou_score: 0.2964
Epoch 10: saving model to model_1_save/unet_m1_best_model_e10.h5
80/80 [==============================] - 15s 187ms/step - loss: 0.6632 - iou_score: 0.2964 - val_loss:
0.8230 - val_iou_score: 0.1734 - lr: 9.0000e-05
Epoch 11/50
80/80 [==============================] - ETA: 0s - loss: 0.6295 - iou_score: 0.3229
Epoch 11: saving model to model_1_save/unet_m1_best_model_e11.h5
80/80 [==============================] - 15s 192ms/step - loss: 0.6295 - iou_score: 0.3229 - val_loss:
0.8004 - val_iou_score: 0.1883 - lr: 9.0000e-05
Epoch 12/50
80/80 [==============================] - ETA: 0s - loss: 0.5955 - iou_score: 0.3513
Epoch 12: saving model to model_1_save/unet_m1_best_model_e12.h5

Epoch 12: ReduceLROnPlateau reducing learning rate to 8.100000122794882e-05.
80/80 [==============================] - 15s 189ms/step - loss: 0.5955 - iou_score: 0.3513 - val_loss:
0.7655 - val_iou_score: 0.2127 - lr: 9.0000e-05
Epoch 13/50
80/80 [==============================] - ETA: 0s - loss: 0.5646 - iou_score: 0.3774
Epoch 13: saving model to model_1_save/unet_m1_best_model_e13.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.5646 - iou_score: 0.3774 - val_loss:
0.7600 - val_iou_score: 0.2179 - lr: 8.1000e-05
Epoch 14/50
80/80 [==============================] - ETA: 0s - loss: 0.5386 - iou_score: 0.4010
Epoch 14: saving model to model_1_save/unet_m1_best_model_e14.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.5386 - iou_score: 0.4010 - val_loss:
0.7369 - val_iou_score: 0.2349 - lr: 8.1000e-05
Epoch 15/50
80/80 [==============================] - ETA: 0s - loss: 0.5173 - iou_score: 0.4216
Epoch 15: saving model to model_1_save/unet_m1_best_model_e15.h5
80/80 [==============================] - 15s 187ms/step - loss: 0.5173 - iou_score: 0.4216 - val_loss:
0.7250 - val_iou_score: 0.2456 - lr: 8.1000e-05
Epoch 16/50
80/80 [==============================] - ETA: 0s - loss: 0.4997 - iou_score: 0.4390
Epoch 16: saving model to model_1_save/unet_m1_best_model_e16.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.4997 - iou_score: 0.4390 - val_loss:
0.7313 - val_iou_score: 0.2433 - lr: 8.1000e-05
```

```
Epoch 17/50
80/80 [==============================] - ETA: 0s - loss: 0.4867 - iou_score: 0.4519
Epoch 17: saving model to model_1_save/unet_m1_best_model_e17.h5

Epoch 17: ReduceLROnPlateau reducing learning rate to 7.289999848580919e-05.
80/80 [==============================] - 15s 191ms/step - loss: 0.4867 - iou_score: 0.4519 - val_loss:
0.7176 - val_iou_score: 0.2557 - lr: 8.1000e-05
Epoch 18/50
80/80 [==============================] - ETA: 0s - loss: 0.4749 - iou_score: 0.4636
Epoch 18: saving model to model_1_save/unet_m1_best_model_e18.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.4749 - iou_score: 0.4636 - val_loss:
0.6967 - val_iou_score: 0.2707 - lr: 7.2900e-05
Epoch 19/50
80/80 [==============================] - ETA: 0s - loss: 0.4632 - iou_score: 0.4742
Epoch 19: saving model to model_1_save/unet_m1_best_model_e19.h5
80/80 [==============================] - 15s 190ms/step - loss: 0.4632 - iou_score: 0.4742 - val_loss:
0.7065 - val_iou_score: 0.2654 - lr: 7.2900e-05
Epoch 20/50
80/80 [==============================] - ETA: 0s - loss: 0.4517 - iou_score: 0.4833
Epoch 20: saving model to model_1_save/unet_m1_best_model_e20.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.4517 - iou_score: 0.4833 - val_loss:
0.6921 - val_iou_score: 0.2762 - lr: 7.2900e-05
Epoch 21/50
80/80 [==============================] - ETA: 0s - loss: 0.4397 - iou_score: 0.4933
Epoch 21: saving model to model_1_save/unet_m1_best_model_e21.h5
80/80 [==============================] - 15s 195ms/step - loss: 0.4397 - iou_score: 0.4933 - val_loss:
0.6842 - val_iou_score: 0.2825 - lr: 7.2900e-05
Epoch 22/50
80/80 [==============================] - ETA: 0s - loss: 0.4285 - iou_score: 0.5029
Epoch 22: saving model to model_1_save/unet_m1_best_model_e22.h5

Epoch 22: ReduceLROnPlateau reducing learning rate to 6.561000191140920e-05.
80/80 [==============================] - 15s 188ms/step - loss: 0.4285 - iou_score: 0.5029 - val_loss:
0.6865 - val_iou_score: 0.2808 - lr: 7.2900e-05
Epoch 23/50
80/80 [==============================] - ETA: 0s - loss: 0.4158 - iou_score: 0.5150
Epoch 23: saving model to model_1_save/unet_m1_best_model_e23.h5
80/80 [==============================] - 15s 192ms/step - loss: 0.4158 - iou_score: 0.5150 - val_loss:
0.6855 - val_iou_score: 0.2822 - lr: 6.5610e-05
Epoch 24/50
80/80 [==============================] - ETA: 0s - loss: 0.4023 - iou_score: 0.5274
Epoch 24: saving model to model_1_save/unet_m1_best_model_e24.h5
80/80 [==============================] - 15s 188ms/step - loss: 0.4023 - iou_score: 0.5274 - val_loss:
0.6730 - val_iou_score: 0.2913 - lr: 6.5610e-05
Epoch 25/50
80/80 [==============================] - ETA: 0s - loss: 0.3905 - iou_score: 0.5371
Epoch 25: saving model to model_1_save/unet_m1_best_model_e25.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.3905 - iou_score: 0.5371 - val_loss:
0.6708 - val_iou_score: 0.2933 - lr: 6.5610e-05
Epoch 26/50
80/80 [==============================] - ETA: 0s - loss: 0.3721 - iou_score: 0.5516
Epoch 26: saving model to model_1_save/unet_m1_best_model_e26.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.3721 - iou_score: 0.5516 - val_loss:
0.6549 - val_iou_score: 0.3038 - lr: 6.5610e-05
Epoch 27/50
80/80 [==============================] - ETA: 0s - loss: 0.3566 - iou_score: 0.5651
Epoch 27: saving model to model_1_save/unet_m1_best_model_e27.h5

Epoch 27: ReduceLROnPlateau reducing learning rate to 5.904900172026828e-05.
80/80 [==============================] - 15s 191ms/step - loss: 0.3566 - iou_score: 0.5651 - val_loss:
0.6471 - val_iou_score: 0.3107 - lr: 6.5610e-05
Epoch 28/50
80/80 [==============================] - ETA: 0s - loss: 0.3458 - iou_score: 0.5762
Epoch 28: saving model to model_1_save/unet_m1_best_model_e28.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.3458 - iou_score: 0.5762 - val_loss:
0.6407 - val_iou_score: 0.3186 - lr: 5.9049e-05
Epoch 29/50
80/80 [==============================] - ETA: 0s - loss: 0.3355 - iou_score: 0.5868
Epoch 29: saving model to model_1_save/unet_m1_best_model_e29.h5
80/80 [==============================] - 15s 190ms/step - loss: 0.3355 - iou_score: 0.5868 - val_loss:
0.6399 - val_iou_score: 0.3197 - lr: 5.9049e-05
Epoch 30/50
80/80 [==============================] - ETA: 0s - loss: 0.3280 - iou_score: 0.5948
Epoch 30: saving model to model_1_save/unet_m1_best_model_e30.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.3280 - iou_score: 0.5948 - val_loss:
0.6348 - val_iou_score: 0.3240 - lr: 5.9049e-05
Epoch 31/50
```

```
80/80 [==============================] - ETA: 0s - loss: 0.3218 - iou_score: 0.6014
Epoch 31: saving model to model_1_save/unet_m1_best_model_e31.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.3218 - iou_score: 0.6014 - val_loss:
0.6415 - val_iou_score: 0.3200 - lr: 5.9049e-05
Epoch 32/50
80/80 [==============================] - ETA: 0s - loss: 0.3169 - iou_score: 0.6064
Epoch 32: saving model to model_1_save/unet_m1_best_model_e32.h5

Epoch 32: ReduceLROnPlateau reducing learning rate to 5.3144101548241450e-05.
80/80 [==============================] - 15s 190ms/step - loss: 0.3169 - iou_score: 0.6064 - val_loss:
0.6281 - val_iou_score: 0.3311 - lr: 5.9049e-05
Epoch 33/50
80/80 [==============================] - ETA: 0s - loss: 0.3107 - iou_score: 0.6132
Epoch 33: saving model to model_1_save/unet_m1_best_model_e33.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.3107 - iou_score: 0.6132 - val_loss:
0.6428 - val_iou_score: 0.3209 - lr: 5.3144e-05
Epoch 34/50
80/80 [==============================] - ETA: 0s - loss: 0.3050 - iou_score: 0.6195
Epoch 34: saving model to model_1_save/unet_m1_best_model_e34.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.3050 - iou_score: 0.6195 - val_loss:
0.6290 - val_iou_score: 0.3310 - lr: 5.3144e-05
Epoch 35/50
80/80 [==============================] - ETA: 0s - loss: 0.3016 - iou_score: 0.6234
Epoch 35: saving model to model_1_save/unet_m1_best_model_e35.h5
80/80 [==============================] - 15s 190ms/step - loss: 0.3016 - iou_score: 0.6234 - val_loss:
0.6387 - val_iou_score: 0.3260 - lr: 5.3144e-05
Epoch 36/50
80/80 [==============================] - ETA: 0s - loss: 0.2981 - iou_score: 0.6273
Epoch 36: saving model to model_1_save/unet_m1_best_model_e36.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.2981 - iou_score: 0.6273 - val_loss:
0.6318 - val_iou_score: 0.3307 - lr: 5.3144e-05
Epoch 37/50
80/80 [==============================] - ETA: 0s - loss: 0.2952 - iou_score: 0.6302
Epoch 37: saving model to model_1_save/unet_m1_best_model_e37.h5

Epoch 37: ReduceLROnPlateau reducing learning rate to 4.7829690083744934e-05.
80/80 [==============================] - 15s 192ms/step - loss: 0.2952 - iou_score: 0.6302 - val_loss:
0.6293 - val_iou_score: 0.3324 - lr: 5.3144e-05
Epoch 38/50
80/80 [==============================] - ETA: 0s - loss: 0.2923 - iou_score: 0.6335
Epoch 38: saving model to model_1_save/unet_m1_best_model_e38.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.2923 - iou_score: 0.6335 - val_loss:
0.6324 - val_iou_score: 0.3308 - lr: 4.7830e-05
Epoch 39/50
80/80 [==============================] - ETA: 0s - loss: 0.2885 - iou_score: 0.6375
Epoch 39: saving model to model_1_save/unet_m1_best_model_e39.h5
80/80 [==============================] - 15s 193ms/step - loss: 0.2885 - iou_score: 0.6375 - val_loss:
0.6303 - val_iou_score: 0.3327 - lr: 4.7830e-05
Epoch 40/50
80/80 [==============================] - ETA: 0s - loss: 0.2864 - iou_score: 0.6398
Epoch 40: saving model to model_1_save/unet_m1_best_model_e40.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.2864 - iou_score: 0.6398 - val_loss:
0.6308 - val_iou_score: 0.3328 - lr: 4.7830e-05
Epoch 41/50
80/80 [==============================] - ETA: 0s - loss: 0.2841 - iou_score: 0.6424
Epoch 41: saving model to model_1_save/unet_m1_best_model_e41.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.2841 - iou_score: 0.6424 - val_loss:
0.6241 - val_iou_score: 0.3372 - lr: 4.7830e-05
Epoch 42/50
80/80 [==============================] - ETA: 0s - loss: 0.2829 - iou_score: 0.6437
Epoch 42: saving model to model_1_save/unet_m1_best_model_e42.h5

Epoch 42: ReduceLROnPlateau reducing learning rate to 4.3046720747952350e-05.
80/80 [==============================] - 15s 189ms/step - loss: 0.2829 - iou_score: 0.6437 - val_loss:
0.6362 - val_iou_score: 0.3298 - lr: 4.7830e-05
Epoch 43/50
80/80 [==============================] - ETA: 0s - loss: 0.2801 - iou_score: 0.6470
Epoch 43: saving model to model_1_save/unet_m1_best_model_e43.h5
80/80 [==============================] - 15s 192ms/step - loss: 0.2801 - iou_score: 0.6470 - val_loss:
0.6249 - val_iou_score: 0.3378 - lr: 4.3047e-05
Epoch 44/50
80/80 [==============================] - ETA: 0s - loss: 0.2768 - iou_score: 0.6507
Epoch 44: saving model to model_1_save/unet_m1_best_model_e44.h5
80/80 [==============================] - 15s 188ms/step - loss: 0.2768 - iou_score: 0.6507 - val_loss:
0.6321 - val_iou_score: 0.3341 - lr: 4.3047e-05
Epoch 45/50
80/80 [==============================] - ETA: 0s - loss: 0.2745 - iou_score: 0.6532
```

```
Epoch 45: saving model to model_1_save/unet_m1_best_model_e45.h5
80/80 [==============================] - 15s 189ms/step - loss: 0.2745 - iou_score: 0.6532 - val_loss:
0.6263 - val_iou_score: 0.3376 - lr: 4.3047e-05
Epoch 46/50
80/80 [==============================] - ETA: 0s - loss: 0.2729 - iou_score: 0.6549
Epoch 46: saving model to model_1_save/unet_m1_best_model_e46.h5
80/80 [==============================] - 17s 208ms/step - loss: 0.2729 - iou_score: 0.6549 - val_loss:
0.6277 - val_iou_score: 0.3381 - lr: 4.3047e-05
Epoch 47/50
80/80 [==============================] - ETA: 0s - loss: 0.2709 - iou_score: 0.6571
Epoch 47: saving model to model_1_save/unet_m1_best_model_e47.h5

Epoch 47: ReduceLROnPlateau reducing learning rate to 3.8742047036066654e-05.
80/80 [==============================] - 19s 234ms/step - loss: 0.2709 - iou_score: 0.6571 - val_loss:
0.6311 - val_iou_score: 0.3351 - lr: 4.3047e-05
Epoch 48/50
80/80 [==============================] - ETA: 0s - loss: 0.2693 - iou_score: 0.6591
Epoch 48: saving model to model_1_save/unet_m1_best_model_e48.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.2693 - iou_score: 0.6591 - val_loss:
0.6267 - val_iou_score: 0.3389 - lr: 3.8742e-05
Epoch 49/50
80/80 [==============================] - ETA: 0s - loss: 0.2662 - iou_score: 0.6623
Epoch 49: saving model to model_1_save/unet_m1_best_model_e49.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.2662 - iou_score: 0.6623 - val_loss:
0.6253 - val_iou_score: 0.3400 - lr: 3.8742e-05
Epoch 50/50
80/80 [==============================] - ETA: 0s - loss: 0.2643 - iou_score: 0.6645
Epoch 50: saving model to model_1_save/unet_m1_best_model_e50.h5
80/80 [==============================] - 15s 191ms/step - loss: 0.2643 - iou_score: 0.6645 - val_loss:
0.6297 - val_iou_score: 0.3376 - lr: 3.8742e-05
Time Taken for training (sec):  789.7975680828094
```

In [ ]:

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

In [ ]:

```
np.argmax(history_m1.history['val_iou_score'])
```

Out[ ]:

```
48
```

Predicting patchs using Best unet_m1 weights

In [ ]:

```
# unet_m1.load_weights('/content/model_1_save/unet_m1_best_model_e49.h5')
```

In [ ]:

```
!wget --header="Host: doc-0s-3o-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" --heade
r="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="Cooki
e: AUTH_82jcsesreiehjbhkrct3c4mrj1raokod_nonce=tc458pqm663mq" --header="Connection: keep-alive" "https:
//doc-0s-3o-docs.googleusercontent.com/docs/securesc/rg90kivf62vcrm9d2s7vb24hsj0c3fo2/eng986mro237n72hh
alla2np13i7fc45/1646371650000/16522560826923149764/00176583124175523585/1Wx4yqfovlkmNKJLCTCiLxua2sUwhJv
RR?e=download&ax=ACxEAsZreQjCTpGIEVcusze2-4RbEuOgUVUbler5lGxwz6QURASZzCJesai6D7fjaSjIXTtBt_tpbQXUVP26Bu
OV2__YjsAYgXC8tKWntn7Onx32dtld9ArGO7W0t8Zz1fL8w8mmMuNkrKTTdzngzcMYMHcKCPau007Zjqi9GDBIX4eM4l0HSRdStmjRn
xCvsviS8CFqqNRpu7jIb1r7XWDpJERQJI6k0hl4nGQNqdBx_LOOD44Q8LkVTS-ZwVcP_nKwKgyv-_dIiouFUO6NK3AibTIYhqyIpCsz
AQ1SjfjasvfD8RfgqcrNj4YHrH9lg0XTquAuv0upo0RPSBxTQ6nsIkQ_SRzigtFs2AljXofsJTgTiGsd1H4RQlH09KJZxZafPH74FtT
K1W28mcwc88HYYzix20NWOcIkBaChsskmGBJcidcaytN0G9pKvjMwWJkVn346xIDcm4z_dK6UUIfayTgfB33QSol8rr8qvkZB4ez-lx
UMzpNzd2yeHWqUTk2T2ibbf3XmgOmaAmmMQc7b2776S9LW6F7_k-UaB-1r6O0RwbhR6sab27Q3iEpno2WBo1C9J2eZCB6RZz8UoPoue
_s4hd8D1pVYs6t5BGw2YmMn_3yqR20rA1WLWgcmkr3zDS56VsyJLe7JhxWkGe9GsGpT70mgzmrADvZQ4BF8&authuser=0&nonce=tc
458pqm663mq&user=00176583124175523585&hash=fu585jmepe9e7dttt9jkfleg4la888ej" -c -O 'unet_m1_best_model_
e49.h5'
```

```
e49.h5
```

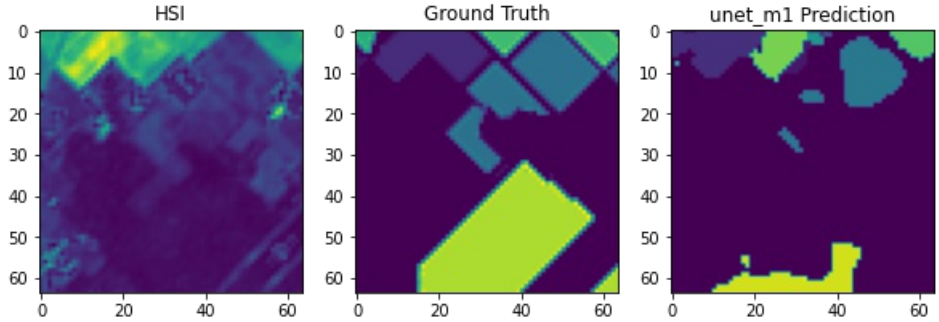
```
--2022-03-04 05:28:04--  https://doc-0s-3o-docs.googleusercontent.com/docs/securesc/rg90kivf62vcrm9d2s7vb24hsj0c3fo2/eng986mro237n72hhalla2np13i7fc45/1646371650000/16522560826923149764/00176583124175523585/1Wx4yqfovlkmNKJLCTCiLxua2sUwhJvRR?e=download&ax=ACxEAsZreQjCTpGIEVcusze2-4RbEuOgUVUbler5lGxwz6QURASZzCJesai6D7fjaSjIXTtBt_tpbQXUVP26BuOV2__YjsAYgXC8tKWntn7Onx32dtld9ArGO7W0t8Zz1fL8w8mmMuNkrKTTdzngzcMYMHcKCPau007Zjqi9GDBIX4eM4l0HSRdStmjRnxCvsviS8CFqqNRpu7jIb1r7XWDpJERQJI6k0hl4nGQNqdBx_LOOD44Q8LkVTS-ZwVcP_nKwKgyv-_dIiouFUO6NK3AibTIYhqyIpCszAQ1SjfjasvfD8RfgqcrNj4YHrH9lg0XTquAuv0upo0RPSBxTQ6nsIkQ_SRzigtFs2AljXofsJTgTiGsd1H4RQlH09KJZxZafPH74FtTK1W28mcwc88HYYzix20NWOcIkBaChsskmGBJcidcaytN0G9pKvjMwWJkVn346xIDcm4z_dK6UUIfayTgfB33QSol8rr8qvkZB4ez-lxUMzpNzd2yeHWqUTk2T2ibbf3XmgOmaAmmMQc7b2776S9LW6F7_k-UaB-1r6O0RwbhR6sab27Q3iEpno2WBo1C9J2eZCB6RZz8UoPoue_s4hd8D1pVYs6t5BGw2YmMn_3yqR20rA1WLWgcmkr3zDS56VsyJLe7JhxWkGe9GsGpT70mgzmrADvZQ4BF8&authuser=0&nonce=tc458pqm663mq&user=00176583124175523585&hash=fu585jmepe9e7dttt9jkfleg4la888ej
Resolving doc-0s-3o-docs.googleusercontent.com (doc-0s-3o-docs.googleusercontent.com)... 142.251.6.132, 2607:f8b0:4001:c5a::84
Connecting to doc-0s-3o-docs.googleusercontent.com (doc-0s-3o-docs.googleusercontent.com)|142.251.6.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 98071376 (94M) [application/octet-stream]
Saving to: 'unet_m1_best_model_e49.h5'

unet_m1_best_model_ 100%[===================>]  93.53M  94.2MB/s    in 1.0s

2022-03-04 05:28:05 (94.2 MB/s) - 'unet_m1_best_model_e49.h5' saved [98071376/98071376]
```

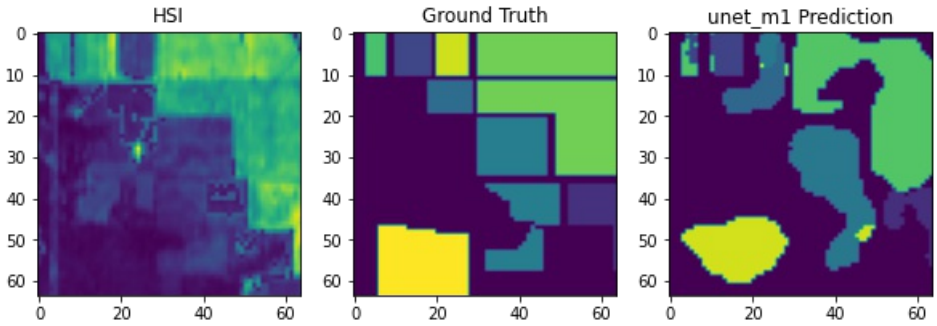
In [ ]:

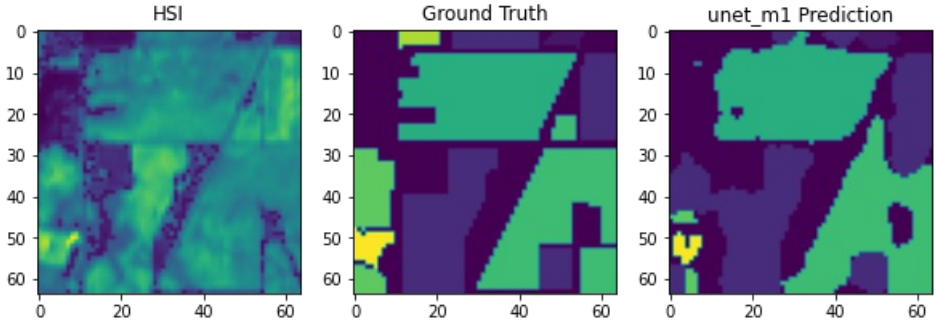
```
# Loading saved model weights
unet_m1.load_weights('unet_m1_best_model_e49.h5')
```

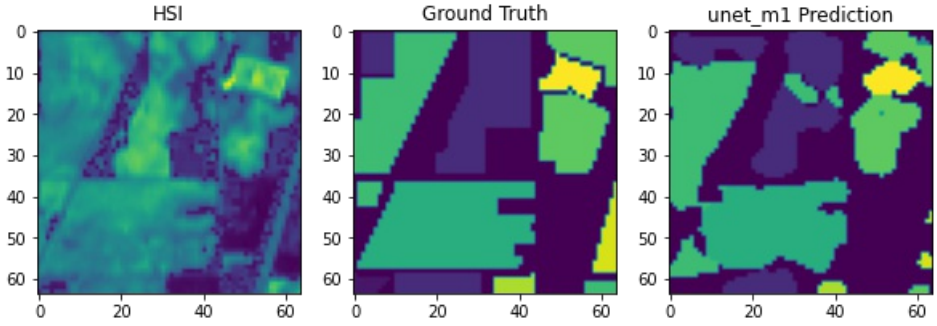
In [ ]:

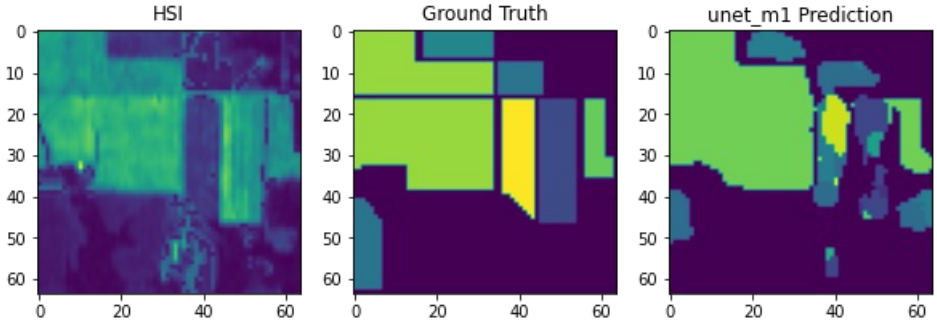
```
# Plotting Model prediction of segmentation alongside HSI and Ground Truth
i=0
for im, gt in zip(X_test[20:100],y_test[20:100]):

    # model prediction
    pred = unet_m1.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_m1 Prediction')
    plt.show()
    i+=1
    if(i>10):
        break
```

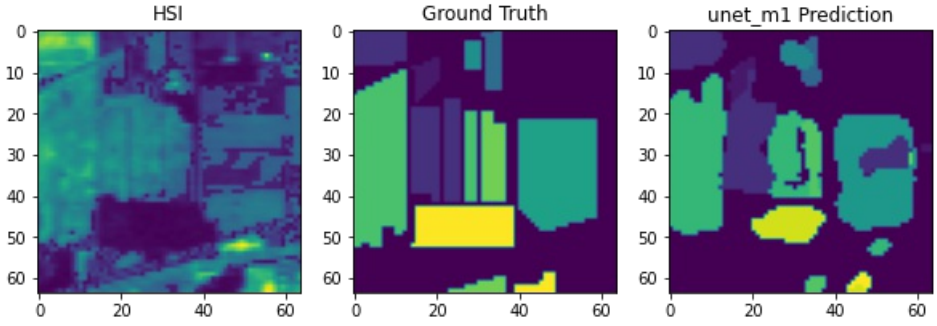



HSI
Ground Truth
unet_m1 Prediction

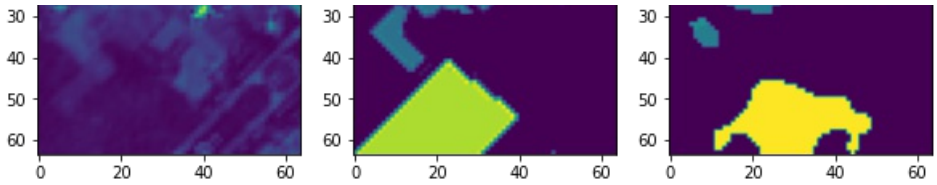

HSI
Ground Truth
unet_m1 Prediction

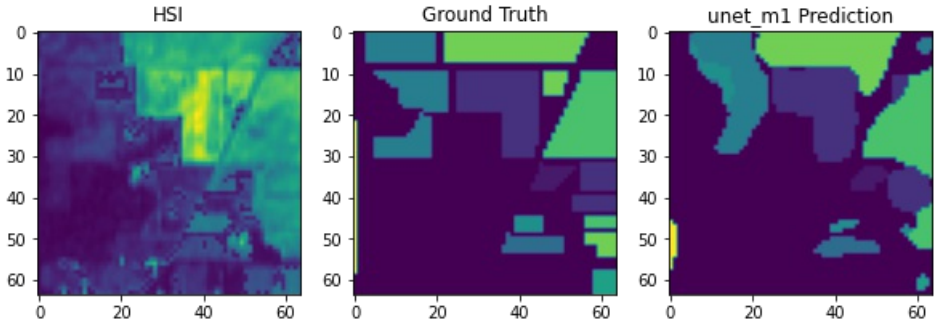

HSI
Ground Truth
unet_m1 Prediction

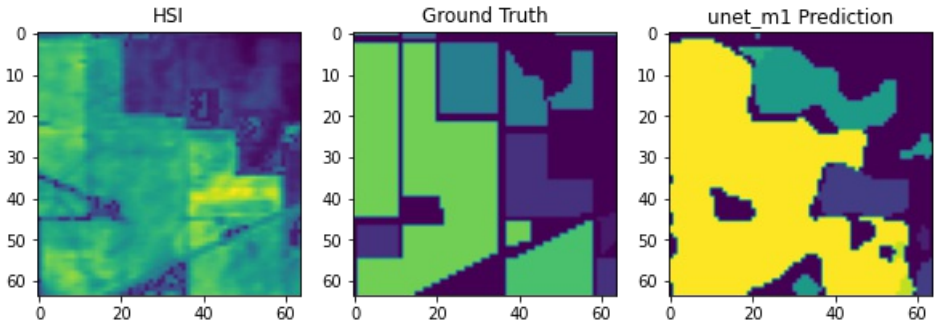

HSI
Ground Truth
unet_m1 Prediction

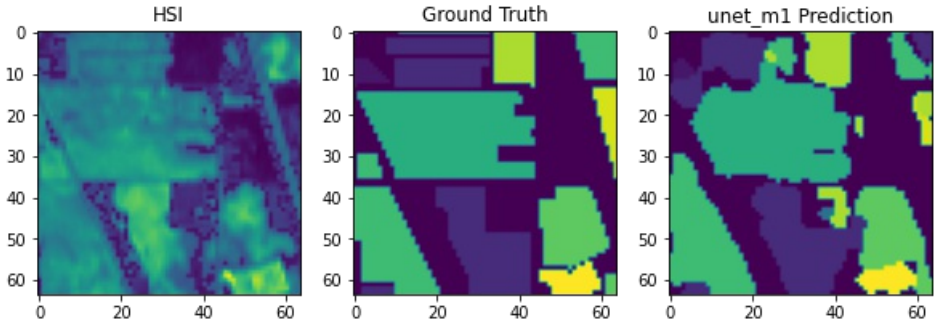

HSI
Ground Truth
unet_m1 Prediction

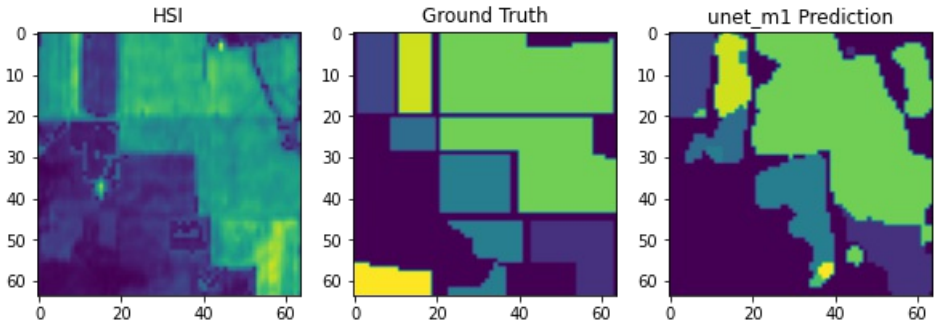

HSI
Ground Truth
unet_m1 Prediction











unet_m1 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_m1.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

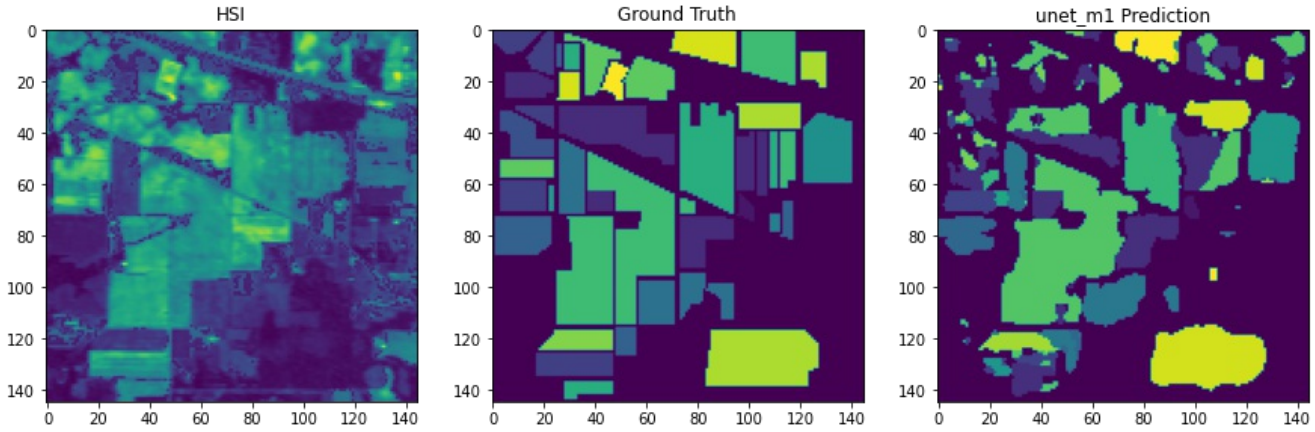
In [ ]:

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

In [ ]:

```
# plottng 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('unet_m1 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 probablities(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
```

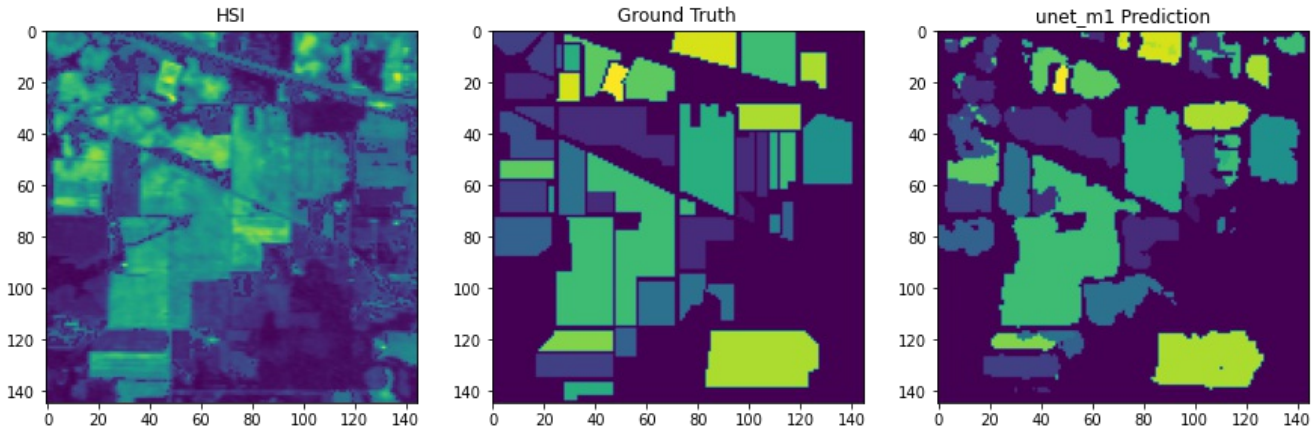
In [ ]:

```
# 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.title('HSI')
plt.subplot(132)
plt.imshow(img_gt)
plt.title('Ground Truth')
plt.subplot(133)
plt.imshow(prediction)
plt.title('unet_m1 Prediction')
plt.show()
```



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

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 [ ]:

```
from sklearn.metrics import f1_score, cohen_kappa_score
```

In [ ]:

```
F1_unet_m1 = f1_score(y,y_hat,average='micro')
print('micro F1 score of pretrained unet model for full image : ',F1_unet_m1)
kappa_unet_m1 = cohen_kappa_score(y,y_hat)
print('kappa score of pretrained unet model for full image : ',kappa_unet_m1)
```

```
micro F1 score of pretrained unet model for full image :  0.8661117717003567
kappa score of pretrained unet model for full image :  0.8029087034442279
```

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_m1.predict(X_test)
prediction_test = np.argmax(pred_test,axis=-1)
```

```
prediction_test = np.argmax(pred_test,axis= 1)
```

In [ ]:

```
prediction_test.shape
```

Out[ ]:

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

In [ ]:

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

In [ ]:

```
F1_unet_m1_val = f1_score(y_val,y_hat_val,average='micro')
print('micro F1 score of pretrained unet model for validation data: ',F1_unet_m1_val)
kappa_unet_m1_val = cohen_kappa_score(y_val,y_hat_val)
print('kappa score of pretrained unet model for validation data: ',kappa_unet_m1_val)
```

```
micro F1 score of pretrained unet model for validation data:  0.72689453125
kappa score of pretrained unet model for validation data:  0.6315991067893165
```

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()
```

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_initializ
er='he_normal')(input)
  Enc_L1 = Conv2D(filters = 64, kernel_size = (3,3), padding='same', activation='relu',kernel_initializ
er='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_initiali
zer='he_normal')(Enc_P1)
  Enc_L2 = Conv2D(filters = 128, kernel_size = (3,3), padding='same', activation='relu',kernel_initiali
zer='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_initiali
zer='he_normal')(Enc_P2)
  Enc_L3 = Conv2D(filters = 256, kernel_size = (3,3), padding='same', activation='relu',kernel_initiali
zer='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_initiali
zer='he_normal')(Enc_P3)
  Enc_L4 = Conv2D(filters = 512, kernel_size = (3,3), padding='same', activation='relu',kernel_initiali
zer='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_initi
alizer='he_normal')(Enc_P4)
  # Enc_L5 = Conv2D(filters = 1024, kernel_size = (3,3), padding='same', activation='relu',kernel_initi
alizer='he_normal')(Enc_L5)

  # Dec_L0 = Conv2DTranspose(filters = 512, kernel_size = (2,2), strides =(2,2), padding='valid')(Enc_L
5)
  # Dec_L0 = concatenate([Dec_L0,Enc_L4])
  # Dec_L0 = Conv2D(filters = 256, kernel_size = (3,3), padding='same', activation='relu',kernel_initia
lizer='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_initiali
zer='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_initiali
zer='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_initializ
er='he_normal')(Dec_L3)

  Dec_L4 = Conv2D(filters = 32, kernel_size = (3,3), padding='same', activation='relu',kernel_initializ
er='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()
```

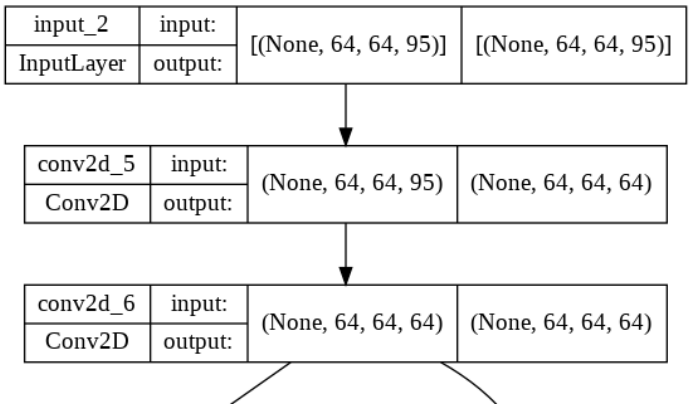
```
Model: "model_2"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 input_2 (InputLayer)        [(None, 64, 64, 95)       0           []
                             ]

 conv2d_5 (Conv2D)           (None, 64, 64, 64)        54784       ['input_2[0][0]']

 conv2d_6 (Conv2D)           (None, 64, 64, 64)        36928       ['conv2d_5[0][0]']
```

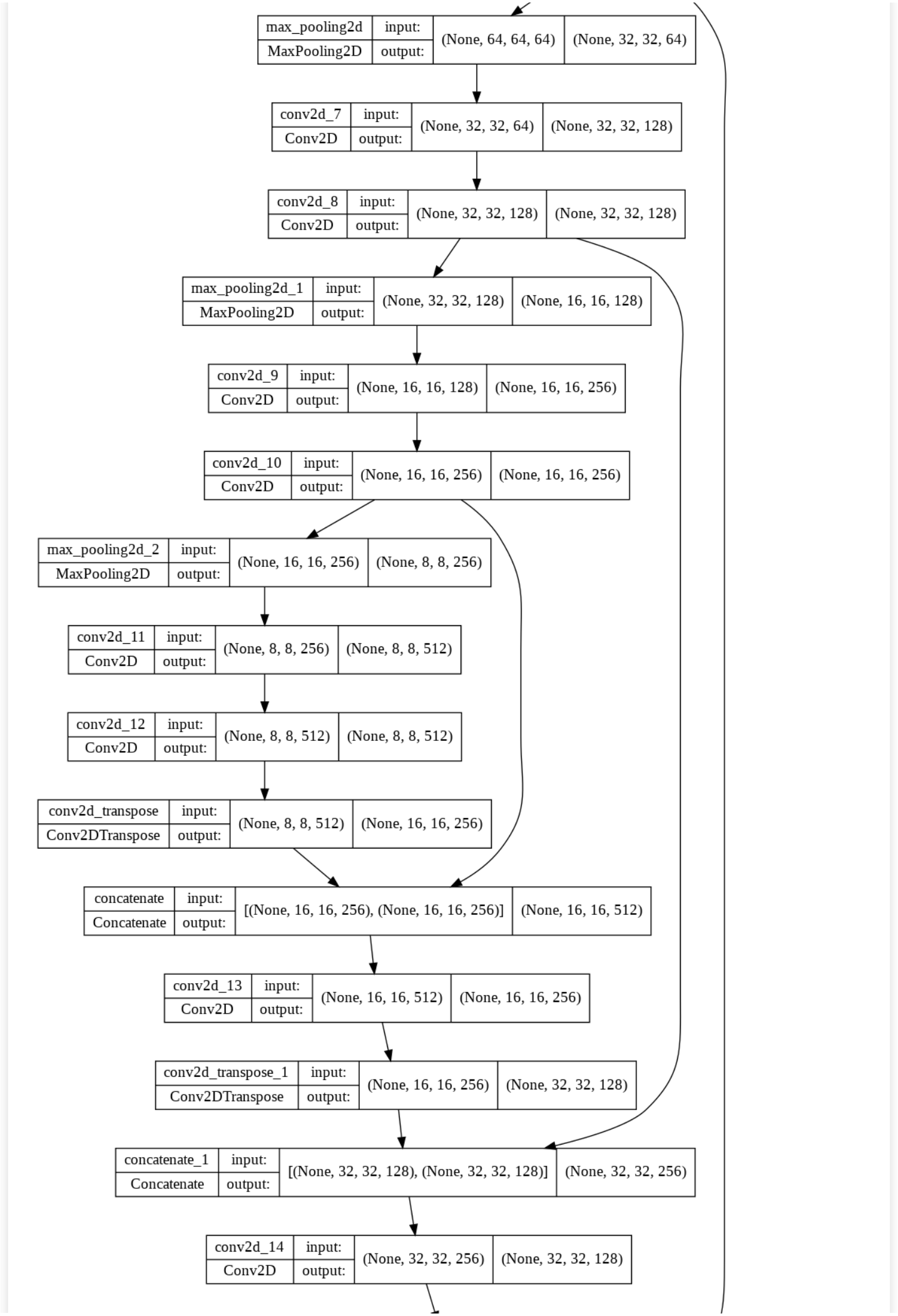
```
max_pooling2d (MaxPooling2D)    (None, 32, 32, 64)    0           ['conv2d_6[0][0]']

conv2d_7 (Conv2D)               (None, 32, 32, 128)   73856       ['max_pooling2d[0][0]']

conv2d_8 (Conv2D)               (None, 32, 32, 128)   147584      ['conv2d_7[0][0]']

max_pooling2d_1 (MaxPooling2D)  (None, 16, 16, 128)   0           ['conv2d_8[0][0]']

conv2d_9 (Conv2D)               (None, 16, 16, 256)   295168      ['max_pooling2d_1[0][0]']

conv2d_10 (Conv2D)              (None, 16, 16, 256)   590080      ['conv2d_9[0][0]']

max_pooling2d_2 (MaxPooling2D)  (None, 8, 8, 256)     0           ['conv2d_10[0][0]']

conv2d_11 (Conv2D)              (None, 8, 8, 512)     1180160     ['max_pooling2d_2[0][0]']

conv2d_12 (Conv2D)              (None, 8, 8, 512)     2359808     ['conv2d_11[0][0]']

conv2d_transpose (Conv2DTranspose)  (None, 16, 16, 256)  524544   ['conv2d_12[0][0]']

concatenate (Concatenate)       (None, 16, 16, 512)   0           ['conv2d_transpose[0][0]',
                                                                   'conv2d_10[0][0]']

conv2d_13 (Conv2D)              (None, 16, 16, 256)   1179904     ['concatenate[0][0]']

conv2d_transpose_1 (Conv2DTranspose)  (None, 32, 32, 128)  131200  ['conv2d_13[0][0]']

concatenate_1 (Concatenate)     (None, 32, 32, 256)   0           ['conv2d_transpose_1[0][0]',
                                                                   'conv2d_8[0][0]']

conv2d_14 (Conv2D)              (None, 32, 32, 128)   295040      ['concatenate_1[0][0]']

conv2d_transpose_2 (Conv2DTranspose)  (None, 64, 64, 64)  32832   ['conv2d_14[0][0]']

concatenate_2 (Concatenate)     (None, 64, 64, 128)   0           ['conv2d_transpose_2[0][0]',
                                                                   'conv2d_6[0][0]']

conv2d_15 (Conv2D)              (None, 64, 64, 64)    73792       ['concatenate_2[0][0]']

conv2d_16 (Conv2D)              (None, 64, 64, 32)    18464       ['conv2d_15[0][0]']

conv2d_17 (Conv2D)              (None, 64, 64, 17)    561         ['conv2d_16[0][0]']

==================================================================================================
Total params: 6,994,705
Trainable params: 6,994,705
Non-trainable params: 0
__________________________________________________________________________________________________
```

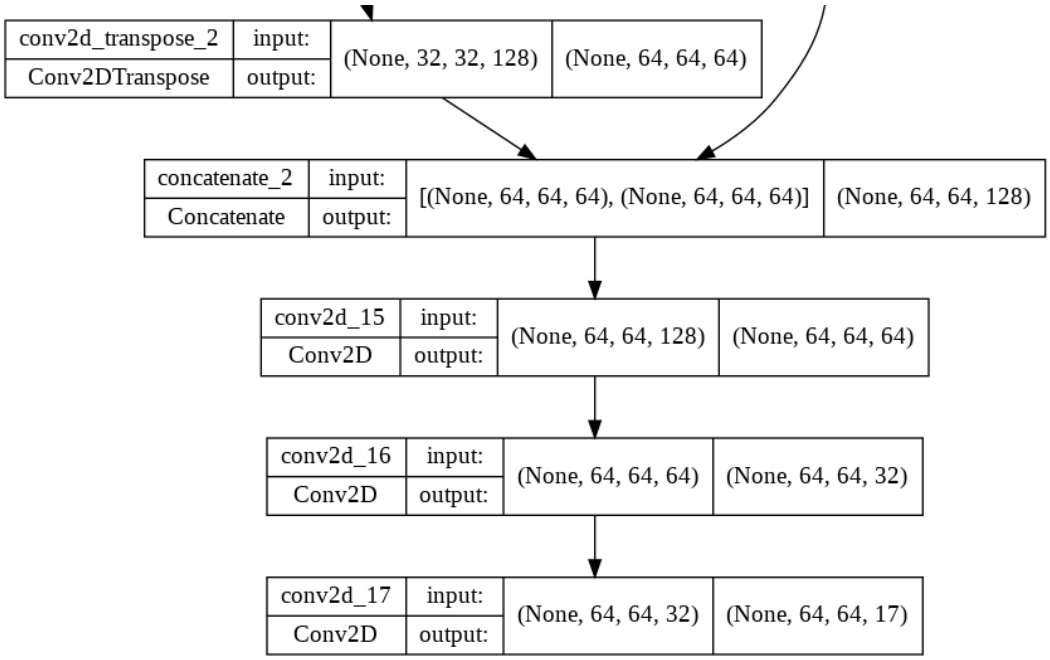
In [ ]:

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

Out[ ]:




max_pooling2d | input: | (None, 64, 64, 64) | (None, 32, 32, 64)
MaxPooling2D | output:
conv2d_7 | input: | (None, 32, 32, 64) | (None, 32, 32, 128)
Conv2D | output:
conv2d_8 | input: | (None, 32, 32, 128) | (None, 32, 32, 128)
Conv2D | output:
max_pooling2d_1 | input: | (None, 32, 32, 128) | (None, 16, 16, 128)
MaxPooling2D | output:
conv2d_9 | input: | (None, 16, 16, 128) | (None, 16, 16, 256)
Conv2D | output:
conv2d_10 | input: | (None, 16, 16, 256) | (None, 16, 16, 256)
Conv2D | output:
max_pooling2d_2 | input: | (None, 16, 16, 256) | (None, 8, 8, 256)
MaxPooling2D | output:
conv2d_11 | input: | (None, 8, 8, 256) | (None, 8, 8, 512)
Conv2D | output:
conv2d_12 | input: | (None, 8, 8, 512) | (None, 8, 8, 512)
Conv2D | output:
conv2d_transpose | input: | (None, 8, 8, 512) | (None, 16, 16, 256)
Conv2DTranspose | output:
concatenate | input: | [(None, 16, 16, 256), (None, 16, 16, 256)] | (None, 16, 16, 512)
Concatenate | output:
conv2d_13 | input: | (None, 16, 16, 512) | (None, 16, 16, 256)
Conv2D | output:
conv2d_transpose_1 | input: | (None, 16, 16, 256) | (None, 32, 32, 128)
Conv2DTranspose | output:
concatenate_1 | input: | [(None, 32, 32, 128), (None, 32, 32, 128)] | (None, 32, 32, 256)
Concatenate | output:
conv2d_14 | input: | (None, 32, 32, 256) | (None, 32, 32, 128)
Conv2D | 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

20220303-114602 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:20: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators. Epoch 1/50 80/80 [==============================] - ETA: 0s - loss: 0.9843 - iou_score: 0.0800 Epoch 1: saving model to model_2_save/unet_m2_best_model_e01.h5 80/80 [==============================] - 24s 122ms/step - loss: 0.9843 - iou_score: 0.0800 - val_loss: 0.8745 - val_iou_score: 0.1488 - lr: 1.0000e-04

Epoch 2/50 80/80 [==============================] - ETA: 0s - loss: 0.8011 - iou_score: 0.2067 Epoch 2: saving model to model_2_save/unet_m2_best_model_e02.h5 80/80 [==============================] - 9s 114ms/step - loss: 0.8011 - iou_score: 0.2067 - val_loss: 0.7253 - val_iou_score: 0.2675 - lr: 1.0000e-04

Epoch 3/50 80/80 [==============================] - ETA: 0s - loss: 0.6608 - iou_score: 0.3145 Epoch 3: saving model to model_2_save/unet_m2_best_model_e03.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.6608 - iou_score: 0.3145 - val_loss: 0.6093 - val_iou_score: 0.3549 - lr: 1.0000e-04

Epoch 4/50 80/80 [==============================] - ETA: 0s - loss: 0.5768 - iou_score: 0.3834 Epoch 4: saving model to model_2_save/unet_m2_best_model_e04.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.5768 - iou_score: 0.3834 - val_loss: 0.5627 - val_iou_score: 0.3929 - lr: 1.0000e-04

Epoch 5/50 80/80 [==============================] - ETA: 0s - loss: 0.4930 - iou_score: 0.4503 Epoch 5: saving model to model_2_save/unet_m2_best_model_e05.h5 80/80 [==============================] - 9s 112ms/step - loss: 0.4930 - iou_score: 0.4503 - val_loss: 0.4639 - val_iou_score: 0.4786 - lr: 1.0000e-04

Epoch 6/50 80/80 [==============================] - ETA: 0s - loss: 0.4403 - iou_score: 0.4991 Epoch 6: 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 113ms/step - loss: 0.4403 - iou_score: 0.4991 - val_loss: 0.4264 - val_iou_score: 0.5095 - lr: 1.0000e-04

Epoch 7/50 80/80 [==============================] - ETA: 0s - loss: 0.4070 - iou_score: 0.5302 Epoch 7: saving model to model_2_save/unet_m2_best_model_e07.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.4070 -

iou_score: 0.5302 - val_loss: 0.3880 - val_iou_score: 0.5482 - lr: 9.0000e-05

Epoch 8/50 80/80 [==============================] - ETA: 0s - loss: 0.3784 - iou_score: 0.5588 Epoch 8: saving model to model_2_save/unet_m2_best_model_e08.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.3784 - iou_score: 0.5588 - val_loss: 0.3646 - val_iou_score: 0.5744 - lr: 9.0000e-05

Epoch 9/50 80/80 [==============================] - ETA: 0s - loss: 0.3539 - iou_score: 0.5845 Epoch 9: saving model to model_2_save/unet_m2_best_model_e09.h5 80/80 [==============================] - 9s 115ms/step - loss: 0.3539 - iou_score: 0.5845 - val_loss: 0.3433 - val_iou_score: 0.5932 - lr: 9.0000e-05

Epoch 10/50 80/80 [==============================] - ETA: 0s - loss: 0.3415 - iou_score: 0.5971 Epoch 10: saving model to model_2_save/unet_m2_best_model_e10.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.3415 - iou_score: 0.5971 - val_loss: 0.3340 - val_iou_score: 0.6023 - lr: 9.0000e-05

Epoch 11/50 80/80 [==============================] - ETA: 0s - loss: 0.3265 - iou_score: 0.6129 Epoch 11: 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 113ms/step - loss: 0.3265 - iou_score: 0.6129 - val_loss: 0.3207 - val_iou_score: 0.6170 - lr: 9.0000e-05

Epoch 12/50 80/80 [==============================] - ETA: 0s - loss: 0.3142 - iou_score: 0.6264 Epoch 12: saving model to model_2_save/unet_m2_best_model_e12.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.3142 - iou_score: 0.6264 - val_loss: 0.3070 - val_iou_score: 0.6322 - lr: 8.1000e-05

Epoch 13/50 80/80 [==============================] - ETA: 0s - loss: 0.3065 - iou_score: 0.6346 Epoch 13: saving model to model_2_save/unet_m2_best_model_e13.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.3065 - iou_score: 0.6346 - val_loss: 0.3042 - val_iou_score: 0.6313 - lr: 8.1000e-05

Epoch 14/50 80/80 [==============================] - ETA: 0s - loss: 0.2994 - iou_score: 0.6413 Epoch 14: saving model to model_2_save/unet_m2_best_model_e14.h5 80/80 [==============================] - 9s 115ms/step - loss: 0.2994 - iou_score: 0.6413 - val_loss: 0.2952 - val_iou_score: 0.6447 - lr: 8.1000e-05

Epoch 15/50 80/80 [==============================] - ETA: 0s - loss: 0.2935 - iou_score: 0.6470 Epoch 15: saving model to model_2_save/unet_m2_best_model_e15.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.2935 - iou_score: 0.6470 - val_loss: 0.2932 - val_iou_score: 0.6458 - lr: 8.1000e-05

Epoch 16/50 80/80 [==============================] - ETA: 0s - loss: 0.2847 - iou_score: 0.6564 Epoch 16: 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 113ms/step - loss: 0.2847 - iou_score: 0.6564 - val_loss: 0.2814 - val_iou_score: 0.6580 - lr: 8.1000e-05

Epoch 17/50 80/80 [==============================] - ETA: 0s - loss: 0.2757 - iou_score: 0.6642 Epoch 17: saving model to model_2_save/unet_m2_best_model_e17.h5 80/80 [==============================] - 10s 121ms/step - loss: 0.2757 - iou_score: 0.6642 - val_loss: 0.2719 - val_iou_score: 0.6657 - lr: 7.2900e-05

Epoch 18/50 80/80 [==============================] - ETA: 0s - loss: 0.2648 - iou_score: 0.6730 Epoch 18: saving model to model_2_save/unet_m2_best_model_e18.h5 80/80 [==============================] - 11s 138ms/step - loss: 0.2648 - iou_score: 0.6730 - val_loss: 0.2661 - val_iou_score: 0.6704 - lr: 7.2900e-05

Epoch 19/50 80/80 [==============================] - ETA: 0s - loss: 0.2529 - iou_score: 0.6839 Epoch 19: saving model to model_2_save/unet_m2_best_model_e19.h5 80/80 [==============================] - 10s 126ms/step - loss: 0.2529 - iou_score: 0.6839 - val_loss: 0.2514 - val_iou_score: 0.6851 - lr: 7.2900e-05

Epoch 20/50 80/80 [==============================] - ETA: 0s - loss: 0.2433 - iou_score: 0.6931 Epoch 20: saving model to model_2_save/unet_m2_best_model_e20.h5 80/80 [==============================] - 9s 114ms/step - loss: 0.2433 - iou_score: 0.6931 - val_loss: 0.2530 - val_iou_score: 0.6822 - lr: 7.2900e-05

Epoch 21/50 80/80 [==============================] - ETA: 0s - loss: 0.2419 - iou_score: 0.6942 Epoch 21: saving model to model_2_save/unet_m2_best_model_e21.h5

Epoch 21: ReduceLROnPlateau reducing learning rate to 6.56100019114092e-05. 80/80 [==============================] - 10s 121ms/step - loss: 0.2419 - iou_score: 0.6942 - val_loss: 0.2478 - val_iou_score: 0.6889 - lr: 7.2900e-05

Epoch 22/50 80/80 [==============================] - ETA: 0s - loss: 0.2297 - iou_score: 0.7072 Epoch 22: saving model to model_2_save/unet_m2_best_model_e22.h5 80/80 [==============================] - 10s 123ms/step - loss: 0.2297 - iou_score: 0.7072 - val_loss: 0.2326 - val_iou_score: 0.7055 - lr: 6.5610e-05

Epoch 23/50 80/80 [==============================] - ETA: 0s - loss: 0.2204 - iou_score: 0.7168 Epoch 23: saving model to model_2_save/unet_m2_best_model_e23.h5 80/80 [==============================] - 9s 116ms/step - loss: 0.2204 - iou_score: 0.7168 - val_loss: 0.2456 - val_iou_score: 0.6902 - lr: 6.5610e-05

Epoch 24/50 80/80 [==============================] - ETA: 0s - loss: 0.2188 - iou_score: 0.7181 Epoch 24: saving model to model_2_save/unet_m2_best_model_e24.h5 80/80 [==============================] - 10s 126ms/step - loss: 0.2188 - iou_score: 0.7181 - val_loss: 0.2221 - val_iou_score: 0.7147 - lr: 6. 5610e-05

Epoch 25/50 80/80 [==============================] - ETA: 0s - loss: 0.2090 - iou_score: 0.7299 Epoch 25: saving model to model_2_save/unet_m2_best_model_e25.h5 80/80 [==============================] - 9s 114ms/step - loss: 0.2090 - iou_score: 0.7299 - val_loss: 0.2203 - val_iou_score: 0.7207 - lr: 6.5610e-05

Epoch 26/50 80/80 [==============================] - ETA: 0s - loss: 0.2086 - iou_score: 0.7298 Epoch 26: 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 113ms/step - loss: 0.2086 - iou_score: 0.7298 - val_loss: 0.2257 - val_iou_score: 0.7110 - lr: 6.5610e-05

Epoch 27/50 80/80 [==============================] - ETA: 0s - loss: 0.2040 - iou_score: 0.7349 Epoch 27: saving model to model_2_save/unet_m2_best_model_e27.h5 80/80 [==============================] - 10s 126ms/step - loss: 0.2040 - iou_score: 0.7349 - val_loss: 0.2161 - val_iou_score: 0.7232 - lr: 5.9049e-05

Epoch 28/50 80/80 [==============================] - ETA: 0s - loss: 0.1983 - iou_score: 0.7415

Epoch 28: saving model to model_2_save/unet_m2_best_model_e28.h5 80/80 [==============================] - 12s 149ms/step - loss: 0.1983 - iou_score: 0.7415 - val_loss: 0.2173 - val_iou_score: 0.7209 - lr: 5.9049e-05

Epoch 29/50 80/80 [==============================] - ETA: 0s - loss: 0.1947 - iou_score: 0.7456 Epoch 29: saving model to model_2_save/unet_m2_best_model_e29.h5 80/80 [==============================] - 9s 115ms/step - loss: 0.1947 - iou_score: 0.7456 - val_loss: 0.2065 - val_iou_score: 0.7346 - lr: 5.9049e-05

Epoch 30/50 80/80 [==============================] - ETA: 0s - loss: 0.1910 - iou_score: 0.7499 Epoch 30: saving model to model_2_save/unet_m2_best_model_e30.h5 80/80 [==============================] - 9s 115ms/step - loss: 0.1910 - iou_score: 0.7499 - val_loss: 0.2086 - val_iou_score: 0.7322 - lr: 5.9049e-05

Epoch 31/50 80/80 [==============================] - ETA: 0s - loss: 0.1888 - iou_score: 0.7519 Epoch 31: saving model to model_2_save/unet_m2_best_model_e31.h5

Epoch 31: ReduceLROnPlateau reducing learning rate to 5.314410154824145e-05. 80/80 [==============================] - 9s 115ms/step - loss: 0.1888 - iou_score: 0.7519 - val_loss: 0.2149 - val_iou_score: 0.7213 - lr: 5.9049e-05

Epoch 32/50 80/80 [==============================] - ETA: 0s - loss: 0.1852 - iou_score: 0.7560 Epoch 32: saving model to model_2_save/unet_m2_best_model_e32.h5 80/80 [==============================] - 10s 121ms/step - loss: 0.1852 - iou_score: 0.7560 - val_loss: 0.1974 - val_iou_score: 0.7424 - lr: 5.3144e-05

Epoch 33/50 80/80 [==============================] - ETA: 0s - loss: 0.1791 - iou_score: 0.7633 Epoch 33: saving model to model_2_save/unet_m2_best_model_e33.h5 80/80 [==============================] - 12s 147ms/step - loss: 0.1791 - iou_score: 0.7633 - val_loss: 0.1910 - val_iou_score: 0.7502 - lr: 5.3144e-05

Epoch 34/50 80/80 [==============================] - ETA: 0s - loss: 0.1755 - iou_score: 0.7675 Epoch 34: saving model to model_2_save/unet_m2_best_model_e34.h5 80/80 [==============================] - 9s 114ms/step - loss: 0.1755 - iou_score: 0.7675 - val_loss: 0.1929 - val_iou_score: 0.7483 - lr: 5.3144e-05

Epoch 35/50 80/80 [==============================] - ETA: 0s - loss: 0.1709 - iou_score: 0.7727 Epoch 35: saving model to model_2_save/unet_m2_best_model_e35.h5 80/80 [==============================] - 9s 114ms/step - loss: 0.1709 - iou_score: 0.7727 - val_loss: 0.1869 - val_iou_score: 0.7556 - lr: 5.3144e-05

Epoch 36/50 80/80 [==============================] - ETA: 0s - loss: 0.1684 - iou_score: 0.7757 Epoch 36: saving model to model_2_save/unet_m2_best_model_e36.h5

Epoch 36: ReduceLROnPlateau reducing learning rate to 4.7829690083744934e-05. 80/80 [==============================] - 9s 113ms/step - loss: 0.1684 - iou_score: 0.7757 - val_loss: 0.1860 - val_iou_score: 0.7575 - lr: 5.3144e-05

Epoch 37/50 80/80 [==============================] - ETA: 0s - loss: 0.1676 - iou_score: 0.7766 Epoch 37: saving model to model_2_save/unet_m2_best_model_e37.h5 80/80 [==============================] - 9s 114ms/step - loss: 0.1676 - iou_score: 0.7766 - val_loss: 0.1872 - val_iou_score: 0.7563 - lr: 4.7830e-05

Epoch 38/50 80/80 [==============================] - ETA: 0s - loss: 0.1624 - iou_score: 0.7827 Epoch 38: saving model to model_2_save/unet_m2_best_model_e38.h5 80/80 [==============================] - 10s 124ms/step - loss: 0.1624 - iou_score: 0.7827 - val_loss: 0.1805 - val_iou_score: 0.7627 - lr: 4.7830e-05

Epoch 39/50 80/80 [==============================] - ETA: 0s - loss: 0.1620 - iou_score: 0.7828 Epoch 39: saving model to model_2_save/unet_m2_best_model_e39.h5 80/80 [==============================] - 10s 130ms/step - loss: 0.1620 - iou_score: 0.7828 - val_loss: 0.1799 - val_iou_score: 0.7628 - lr: 4.7830e-05

Epoch 40/50 80/80 [==============================] - ETA: 0s - loss: 0.1601 - iou_score: 0.7849 Epoch 40: saving model to model_2_save/unet_m2_best_model_e40.h5 80/80 [==============================] - 9s 115ms/step - loss: 0.1601 - iou_score: 0.7849 - val_loss: 0.1764 - val_iou_score: 0.7669 - lr: 4.7830e-05

Epoch 41/50 80/80 [==============================] - ETA: 0s - loss: 0.1565 - iou_score: 0.7889 Epoch 41: saving model to model_2_save/unet_m2_best_model_e41.h5

Epoch 41: ReduceLROnPlateau reducing learning rate to 4.304672074795235e-05. 80/80 [==============================] - 9s 113ms/step - loss: 0.1565 - iou_score: 0.7889 - val_loss: 0.1782 - val_iou_score: 0.7654 - lr: 4.7830e-05

Epoch 42/50 80/80 [==============================] - ETA: 0s - loss: 0.1529 - iou_score: 0.7935 Epoch 42: saving model to model_2_save/unet_m2_best_model_e42.h5 80/80 [==============================] - 9s 112ms/step - loss: 0.1529 - iou_score: 0.7935 - val_loss: 0.1740 - val_iou_score: 0.7698 - lr: 4.3047e-05

Epoch 43/50 80/80 [==============================] - ETA: 0s - loss: 0.1498 - iou_score: 0.7971 Epoch 43: saving model to model_2_save/unet_m2_best_model_e43.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.1498 - iou_score: 0.7971 - val_loss: 0.1719 - val_iou_score: 0.7713 - lr: 4.3047e-05

Epoch 44/50 80/80 [==============================] - ETA: 0s - loss: 0.1470 - iou_score: 0.8005 Epoch 44: saving model to model_2_save/unet_m2_best_model_e44.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.1470 - iou_score: 0.8005 - val_loss: 0.1727 - val_iou_score: 0.7720 - lr: 4.3047e-05

Epoch 45/50 80/80 [==============================] - ETA: 0s - loss: 0.1463 - iou_score: 0.8012 Epoch 45: saving model to model_2_save/unet_m2_best_model_e45.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.1463 - iou_score: 0.8012 - val_loss: 0.1718 - val_iou_score: 0.7736 - lr: 4.3047e-05

Epoch 46/50 80/80 [==============================] - ETA: 0s - loss: 0.1467 - iou_score: 0.8005 Epoch 46: 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 113ms/step - loss: 0.1467 - iou_score: 0.8005 - val_loss: 0.1713 - val_iou_score: 0.7735 - lr: 4.3047e-05

Epoch 47/50 80/80 [==============================] - ETA: 0s - loss: 0.1435 - iou_score: 0.8045 Epoch 47: saving model to model_2_save/unet_m2_best_model_e47.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.1435 - iou_score: 0.8045 - val_loss: 0.1673 - val_iou_score: 0.7771 - lr: 3.8742e-05

Epoch 48/50 80/80 [==============================] - ETA: 0s - loss: 0.1429 - iou_score: 0.8053 Epoch 48: saving model to model_2_save/unet_m2_best_model_e48.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.1429 - iou_score: 0.8053 - val_loss: 0.1670 - val_iou_score: 0.7791 - lr: 3.8742e-05

Epoch 49/50 80/80 [==============================] - ETA: 0s - loss: 0.1386 - iou_score: 0.8108 Epoch 49: saving model to model_2_save/unet_m2_best_model_e49.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.1386 - iou_score: 0.8108 - val_loss: 0.1668 - val_iou_score: 0.7793 - lr: 3.8742e-05

Epoch 50/50 80/80 [==============================] - ETA: 0s - loss: 0.1366 - iou_score: 0.8132 Epoch 50: saving model to model_2_save/unet_m2_best_model_e50.h5 80/80 [==============================] - 9s 113ms/step - loss: 0.1366 - iou_score: 0.8132 - val_loss: 0.1647 - val_iou_score: 0.7814 - lr: 3.8742e-05 Time Taken for training (sec): 502.0929501056671

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_e50+{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,verbo
se=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)
```

```
20220303-121410
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: 0.1362 - iou_score: 0.8138
Epoch 1: val_iou_score improved from -inf to 0.78607, saving model to model_2_save2/unet_m2_best_model_
e50+01.h5
80/80 [==============================] - 11s 119ms/step - loss: 0.1362 - iou_score: 0.8138 - val_loss:
0.1600 - val_iou_score: 0.7861 - lr: 3.8742e-05
Epoch 2/50
80/80 [==============================] - ETA: 0s - loss: 0.1350 - iou_score: 0.8151
Epoch 2: val_iou_score did not improve from 0.78607
80/80 [==============================] - 9s 112ms/step - loss: 0.1350 - iou_score: 0.8151 - val_loss: 0
.1655 - val_iou_score: 0.7814 - lr: 3.8742e-05
Epoch 3/50
80/80 [==============================] - ETA: 0s - loss: 0.1347 - iou_score: 0.8156
Epoch 3: val_iou_score improved from 0.78607 to 0.78641, saving model to model_2_save2/unet_m2_best_mod
el_e50+03.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1347 - iou_score: 0.8156 - val_loss: 0
.1599 - val_iou_score: 0.7864 - lr: 3.8742e-05
Epoch 4/50
80/80 [==============================] - ETA: 0s - loss: 0.1311 - iou_score: 0.8204
Epoch 4: val_iou_score improved from 0.78641 to 0.78679, saving model to model_2_save2/unet_m2_best_mod
el_e50+04.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1311 - iou_score: 0.8204 - val_loss: 0
.1599 - val_iou_score: 0.7868 - lr: 3.8742e-05
Epoch 5/50
80/80 [==============================] - ETA: 0s - loss: 0.1284 - iou_score: 0.8241
Epoch 5: val_iou_score improved from 0.78679 to 0.78842, saving model to model_2_save2/unet_m2_best_mod
el_e50+05.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1284 - iou_score: 0.8241 - val_loss: 0
.1586 - val_iou_score: 0.7884 - lr: 3.8742e-05
Epoch 6/50
80/80 [==============================] - ETA: 0s - loss: 0.1266 - iou_score: 0.8265
Epoch 6: val_iou_score improved from 0.78842 to 0.79251, saving model to model_2_save2/unet_m2_best_mod
el_e50+06.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1266 - iou_score: 0.8265 - val_loss: 0
.1563 - val_iou_score: 0.7925 - lr: 3.8742e-05
Epoch 7/50
80/80 [==============================] - ETA: 0s - loss: 0.1258 - iou_score: 0.8272
Epoch 7: val_iou_score did not improve from 0.79251

Epoch 7: ReduceLROnPlateau reducing learning rate to 3.486780042294413e-05.
80/80 [==============================] - 9s 112ms/step - loss: 0.1258 - iou_score: 0.8272 - val_loss: 0
.1591 - val_iou_score: 0.7894 - lr: 3.8742e-05
Epoch 8/50
80/80 [==============================] - ETA: 0s - loss: 0.1249 - iou_score: 0.8283
Epoch 8: val_iou_score did not improve from 0.79251
80/80 [==============================] - 9s 113ms/step - loss: 0.1249 - iou_score: 0.8283 - val_loss: 0
.1559 - val_iou_score: 0.7924 - lr: 3.4868e-05
```

```
Epoch 9/50
80/80 [==============================] - ETA: 0s - loss: 0.1237 - iou_score: 0.8301
Epoch 9: val_iou_score improved from 0.79251 to 0.79252, saving model to model_2_save2/unet_m2_best_mod
el_e50+09.h5
80/80 [==============================] - 9s 114ms/step - loss: 0.1237 - iou_score: 0.8301 - val_loss: 0
.1564 - val_iou_score: 0.7925 - lr: 3.4868e-05
Epoch 10/50
80/80 [==============================] - ETA: 0s - loss: 0.1209 - iou_score: 0.8336
Epoch 10: val_iou_score improved from 0.79252 to 0.79668, saving model to model_2_save2/unet_m2_best_mo
del_e50+10.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1209 - iou_score: 0.8336 - val_loss: 0
.1528 - val_iou_score: 0.7967 - lr: 3.4868e-05
Epoch 11/50
80/80 [==============================] - ETA: 0s - loss: 0.1197 - iou_score: 0.8351
Epoch 11: val_iou_score did not improve from 0.79668
80/80 [==============================] - 9s 112ms/step - loss: 0.1197 - iou_score: 0.8351 - val_loss: 0
.1557 - val_iou_score: 0.7939 - lr: 3.4868e-05
Epoch 12/50
80/80 [==============================] - ETA: 0s - loss: 0.1209 - iou_score: 0.8334
Epoch 12: val_iou_score did not improve from 0.79668

Epoch 12: ReduceLROnPlateau reducing learning rate to 3.138102038064972e-05.
80/80 [==============================] - 9s 112ms/step - loss: 0.1209 - iou_score: 0.8334 - val_loss: 0
.1546 - val_iou_score: 0.7945 - lr: 3.4868e-05
Epoch 13/50
80/80 [==============================] - ETA: 0s - loss: 0.1177 - iou_score: 0.8376
Epoch 13: val_iou_score improved from 0.79668 to 0.80021, saving model to model_2_save2/unet_m2_best_mo
del_e50+13.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1177 - iou_score: 0.8376 - val_loss: 0
.1494 - val_iou_score: 0.8002 - lr: 3.1381e-05
Epoch 14/50
80/80 [==============================] - ETA: 0s - loss: 0.1154 - iou_score: 0.8406
Epoch 14: val_iou_score did not improve from 0.80021
80/80 [==============================] - 9s 112ms/step - loss: 0.1154 - iou_score: 0.8406 - val_loss: 0
.1501 - val_iou_score: 0.8002 - lr: 3.1381e-05
Epoch 15/50
80/80 [==============================] - ETA: 0s - loss: 0.1155 - iou_score: 0.8406
Epoch 15: val_iou_score did not improve from 0.80021
80/80 [==============================] - 9s 112ms/step - loss: 0.1155 - iou_score: 0.8406 - val_loss: 0
.1505 - val_iou_score: 0.7999 - lr: 3.1381e-05
Epoch 16/50
80/80 [==============================] - ETA: 0s - loss: 0.1145 - iou_score: 0.8417
Epoch 16: val_iou_score improved from 0.80021 to 0.80176, saving model to model_2_save2/unet_m2_best_mo
del_e50+16.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1145 - iou_score: 0.8417 - val_loss: 0
.1483 - val_iou_score: 0.8018 - lr: 3.1381e-05
Epoch 17/50
80/80 [==============================] - ETA: 0s - loss: 0.1139 - iou_score: 0.8422
Epoch 17: val_iou_score did not improve from 0.80176

Epoch 17: ReduceLROnPlateau reducing learning rate to 2.824291768774856e-05.
80/80 [==============================] - 9s 112ms/step - loss: 0.1139 - iou_score: 0.8422 - val_loss: 0
.1482 - val_iou_score: 0.8013 - lr: 3.1381e-05
Epoch 18/50
80/80 [==============================] - ETA: 0s - loss: 0.1113 - iou_score: 0.8458
Epoch 18: val_iou_score improved from 0.80176 to 0.80417, saving model to model_2_save2/unet_m2_best_mo
del_e50+18.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1113 - iou_score: 0.8458 - val_loss: 0
.1467 - val_iou_score: 0.8042 - lr: 2.8243e-05
Epoch 19/50
80/80 [==============================] - ETA: 0s - loss: 0.1104 - iou_score: 0.8470
Epoch 19: val_iou_score did not improve from 0.80417
80/80 [==============================] - 9s 112ms/step - loss: 0.1104 - iou_score: 0.8470 - val_loss: 0
.1468 - val_iou_score: 0.8041 - lr: 2.8243e-05
Epoch 20/50
80/80 [==============================] - ETA: 0s - loss: 0.1095 - iou_score: 0.8482
Epoch 20: val_iou_score improved from 0.80417 to 0.80552, saving model to model_2_save2/unet_m2_best_mo
del_e50+20.h5
80/80 [==============================] - 9s 112ms/step - loss: 0.1095 - iou_score: 0.8482 - val_loss: 0
.1458 - val_iou_score: 0.8055 - lr: 2.8243e-05
Epoch 21/50
80/80 [==============================] - ETA: 0s - loss: 0.1084 - iou_score: 0.8496
Epoch 21: val_iou_score improved from 0.80552 to 0.80670, saving model to model_2_save2/unet_m2_best_mo
del_e50+21.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1084 - iou_score: 0.8496 - val_loss: 0
.1441 - val_iou_score: 0.8067 - lr: 2.8243e-05
Epoch 22/50
```

```
80/80 [==============================] - ETA: 0s - loss: 0.1074 - iou_score: 0.8509
Epoch 22: val_iou_score improved from 0.80670 to 0.80834, saving model to model_2_save2/unet_m2_best_mo
del_e50+22.h5

Epoch 22: ReduceLROnPlateau reducing learning rate to 2.5418625591555612e-05.
80/80 [==============================] - 9s 114ms/step - loss: 0.1074 - iou_score: 0.8509 - val_loss: 0
.1433 - val_iou_score: 0.8083 - lr: 2.8243e-05
Epoch 23/50
80/80 [==============================] - ETA: 0s - loss: 0.1059 - iou_score: 0.8530
Epoch 23: val_iou_score did not improve from 0.80834
80/80 [==============================] - 9s 112ms/step - loss: 0.1059 - iou_score: 0.8530 - val_loss: 0
.1444 - val_iou_score: 0.8071 - lr: 2.5419e-05
Epoch 24/50
80/80 [==============================] - ETA: 0s - loss: 0.1054 - iou_score: 0.8537
Epoch 24: val_iou_score did not improve from 0.80834
80/80 [==============================] - 9s 113ms/step - loss: 0.1054 - iou_score: 0.8537 - val_loss: 0
.1455 - val_iou_score: 0.8058 - lr: 2.5419e-05
Epoch 25/50
80/80 [==============================] - ETA: 0s - loss: 0.1058 - iou_score: 0.8531
Epoch 25: val_iou_score improved from 0.80834 to 0.80925, saving model to model_2_save2/unet_m2_best_mo
del_e50+25.h5
80/80 [==============================] - 9s 114ms/step - loss: 0.1058 - iou_score: 0.8531 - val_loss: 0
.1424 - val_iou_score: 0.8093 - lr: 2.5419e-05
Epoch 26/50
80/80 [==============================] - ETA: 0s - loss: 0.1041 - iou_score: 0.8553
Epoch 26: val_iou_score improved from 0.80925 to 0.80947, saving model to model_2_save2/unet_m2_best_mo
del_e50+26.h5
80/80 [==============================] - 9s 114ms/step - loss: 0.1041 - iou_score: 0.8553 - val_loss: 0
.1418 - val_iou_score: 0.8095 - lr: 2.5419e-05
Epoch 27/50
80/80 [==============================] - ETA: 0s - loss: 0.1042 - iou_score: 0.8551
Epoch 27: val_iou_score did not improve from 0.80947

Epoch 27: ReduceLROnPlateau reducing learning rate to 2.2876762704981958e-05.
80/80 [==============================] - 9s 113ms/step - loss: 0.1042 - iou_score: 0.8551 - val_loss: 0
.1424 - val_iou_score: 0.8094 - lr: 2.5419e-05
Epoch 28/50
80/80 [==============================] - ETA: 0s - loss: 0.1035 - iou_score: 0.8559
Epoch 28: val_iou_score improved from 0.80947 to 0.81058, saving model to model_2_save2/unet_m2_best_mo
del_e50+28.h5
80/80 [==============================] - 9s 112ms/step - loss: 0.1035 - iou_score: 0.8559 - val_loss: 0
.1416 - val_iou_score: 0.8106 - lr: 2.2877e-05
Epoch 29/50
80/80 [==============================] - ETA: 0s - loss: 0.1019 - iou_score: 0.8582
Epoch 29: val_iou_score did not improve from 0.81058
80/80 [==============================] - 9s 112ms/step - loss: 0.1019 - iou_score: 0.8582 - val_loss: 0
.1435 - val_iou_score: 0.8081 - lr: 2.2877e-05
Epoch 30/50
80/80 [==============================] - ETA: 0s - loss: 0.1015 - iou_score: 0.8586
Epoch 30: val_iou_score improved from 0.81058 to 0.81273, saving model to model_2_save2/unet_m2_best_mo
del_e50+30.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.1015 - iou_score: 0.8586 - val_loss: 0
.1396 - val_iou_score: 0.8127 - lr: 2.2877e-05
Epoch 31/50
80/80 [==============================] - ETA: 0s - loss: 0.1010 - iou_score: 0.8592
Epoch 31: val_iou_score did not improve from 0.81273
80/80 [==============================] - 9s 112ms/step - loss: 0.1010 - iou_score: 0.8592 - val_loss: 0
.1423 - val_iou_score: 0.8095 - lr: 2.2877e-05
Epoch 32/50
80/80 [==============================] - ETA: 0s - loss: 0.1008 - iou_score: 0.8594
Epoch 32: val_iou_score did not improve from 0.81273

Epoch 32: ReduceLROnPlateau reducing learning rate to 2.0589085943356624e-05.
80/80 [==============================] - 9s 113ms/step - loss: 0.1008 - iou_score: 0.8594 - val_loss: 0
.1400 - val_iou_score: 0.8122 - lr: 2.2877e-05
Epoch 33/50
80/80 [==============================] - ETA: 0s - loss: 0.0991 - iou_score: 0.8618
Epoch 33: val_iou_score improved from 0.81273 to 0.81327, saving model to model_2_save2/unet_m2_best_mo
del_e50+33.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.0991 - iou_score: 0.8618 - val_loss: 0
.1392 - val_iou_score: 0.8133 - lr: 2.0589e-05
Epoch 34/50
80/80 [==============================] - ETA: 0s - loss: 0.0983 - iou_score: 0.8629
Epoch 34: val_iou_score did not improve from 0.81327
80/80 [==============================] - 9s 111ms/step - loss: 0.0983 - iou_score: 0.8629 - val_loss: 0
.1400 - val_iou_score: 0.8122 - lr: 2.0589e-05
Epoch 35/50
```

```
80/80 [==============================] - ETA: 0s - loss: 0.0974 - iou_score: 0.8642
Epoch 35: val_iou_score improved from 0.81327 to 0.81460, saving model to model_2_save2/unet_m2_best_mo
del_e50+35.h5
80/80 [==============================] - 9s 112ms/step - loss: 0.0974 - iou_score: 0.8642 - val_loss: 0
.1380 - val_iou_score: 0.8146 - lr: 2.0589e-05
Epoch 36/50
80/80 [==============================] - ETA: 0s - loss: 0.0975 - iou_score: 0.8641
Epoch 36: val_iou_score improved from 0.81460 to 0.81519, saving model to model_2_save2/unet_m2_best_mo
del_e50+36.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.0975 - iou_score: 0.8641 - val_loss: 0
.1375 - val_iou_score: 0.8152 - lr: 2.0589e-05
Epoch 37/50
80/80 [==============================] - ETA: 0s - loss: 0.0963 - iou_score: 0.8657
Epoch 37: val_iou_score improved from 0.81519 to 0.81573, saving model to model_2_save2/unet_m2_best_mo
del_e50+37.h5

Epoch 37: ReduceLROnPlateau reducing learning rate to 1.853017784014810e-05.
80/80 [==============================] - 9s 114ms/step - loss: 0.0963 - iou_score: 0.8657 - val_loss: 0
.1371 - val_iou_score: 0.8157 - lr: 2.0589e-05
Epoch 38/50
80/80 [==============================] - ETA: 0s - loss: 0.0953 - iou_score: 0.8671
Epoch 38: val_iou_score improved from 0.81573 to 0.81607, saving model to model_2_save2/unet_m2_best_mo
del_e50+38.h5
80/80 [==============================] - 9s 114ms/step - loss: 0.0953 - iou_score: 0.8671 - val_loss: 0
.1367 - val_iou_score: 0.8161 - lr: 1.8530e-05
Epoch 39/50
80/80 [==============================] - ETA: 0s - loss: 0.0947 - iou_score: 0.8680
Epoch 39: val_iou_score improved from 0.81607 to 0.81655, saving model to model_2_save2/unet_m2_best_mo
del_e50+39.h5
80/80 [==============================] - 9s 115ms/step - loss: 0.0947 - iou_score: 0.8680 - val_loss: 0
.1366 - val_iou_score: 0.8166 - lr: 1.8530e-05
Epoch 40/50
80/80 [==============================] - ETA: 0s - loss: 0.0942 - iou_score: 0.8686
Epoch 40: val_iou_score did not improve from 0.81655
80/80 [==============================] - 9s 114ms/step - loss: 0.0942 - iou_score: 0.8686 - val_loss: 0
.1365 - val_iou_score: 0.8164 - lr: 1.8530e-05
Epoch 41/50
80/80 [==============================] - ETA: 0s - loss: 0.0934 - iou_score: 0.8697
Epoch 41: val_iou_score did not improve from 0.81655
80/80 [==============================] - 9s 114ms/step - loss: 0.0934 - iou_score: 0.8697 - val_loss: 0
.1367 - val_iou_score: 0.8165 - lr: 1.8530e-05
Epoch 42/50
80/80 [==============================] - ETA: 0s - loss: 0.0930 - iou_score: 0.8702
Epoch 42: val_iou_score did not improve from 0.81655

Epoch 42: ReduceLROnPlateau reducing learning rate to 1.667716005613329e-05.
80/80 [==============================] - 9s 113ms/step - loss: 0.0930 - iou_score: 0.8702 - val_loss: 0
.1370 - val_iou_score: 0.8160 - lr: 1.8530e-05
Epoch 43/50
80/80 [==============================] - ETA: 0s - loss: 0.0934 - iou_score: 0.8695
Epoch 43: val_iou_score did not improve from 0.81655
80/80 [==============================] - 9s 114ms/step - loss: 0.0934 - iou_score: 0.8695 - val_loss: 0
.1367 - val_iou_score: 0.8164 - lr: 1.6677e-05
Epoch 44/50
80/80 [==============================] - ETA: 0s - loss: 0.0930 - iou_score: 0.8701
Epoch 44: val_iou_score did not improve from 0.81655
80/80 [==============================] - 9s 114ms/step - loss: 0.0930 - iou_score: 0.8701 - val_loss: 0
.1365 - val_iou_score: 0.8164 - lr: 1.6677e-05
Epoch 45/50
80/80 [==============================] - ETA: 0s - loss: 0.0925 - iou_score: 0.8708
Epoch 45: val_iou_score did not improve from 0.81655
80/80 [==============================] - 9s 114ms/step - loss: 0.0925 - iou_score: 0.8708 - val_loss: 0
.1385 - val_iou_score: 0.8149 - lr: 1.6677e-05
Epoch 46/50
80/80 [==============================] - ETA: 0s - loss: 0.0925 - iou_score: 0.8707
Epoch 46: val_iou_score improved from 0.81655 to 0.81789, saving model to model_2_save2/unet_m2_best_mo
del_e50+46.h5
80/80 [==============================] - 9s 115ms/step - loss: 0.0925 - iou_score: 0.8707 - val_loss: 0
.1357 - val_iou_score: 0.8179 - lr: 1.6677e-05
Epoch 47/50
80/80 [==============================] - ETA: 0s - loss: 0.0925 - iou_score: 0.8707
Epoch 47: val_iou_score did not improve from 0.81789

Epoch 47: ReduceLROnPlateau reducing learning rate to 1.500944454164710e-05.
80/80 [==============================] - 9s 114ms/step - loss: 0.0925 - iou_score: 0.8707 - val_loss: 0
.1389 - val_iou_score: 0.8148 - lr: 1.6677e-05
Epoch 48/50
```

```
80/80 [==============================] - ETA: 0s - loss: 0.0915 - iou_score: 0.8721
Epoch 48: val_iou_score improved from 0.81789 to 0.81807, saving model to model_2_save2/unet_m2_best_mo
del_e50+48.h5
80/80 [==============================] - 9s 113ms/step - loss: 0.0915 - iou_score: 0.8721 - val_loss: 0
.1354 - val_iou_score: 0.8181 - lr: 1.5009e-05
Epoch 49/50
80/80 [==============================] - ETA: 0s - loss: 0.0905 - iou_score: 0.8735
Epoch 49: val_iou_score improved from 0.81807 to 0.81936, saving model to model_2_save2/unet_m2_best_mo
del_e50+49.h5
80/80 [==============================] - 9s 114ms/step - loss: 0.0905 - iou_score: 0.8735 - val_loss: 0
.1343 - val_iou_score: 0.8194 - lr: 1.5009e-05
Epoch 50/50
80/80 [==============================] - ETA: 0s - loss: 0.0898 - iou_score: 0.8745
Epoch 50: val_iou_score did not improve from 0.81936
80/80 [==============================] - 9s 113ms/step - loss: 0.0898 - iou_score: 0.8745 - val_loss: 0
.1348 - val_iou_score: 0.8189 - lr: 1.5009e-05
Time Taken for training (sec):  459.3442509174347
```

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[ ]:

```
48
```

Predicting patchs using Best unet_m2 weights

In [ ]:

```
unet_m2.load_weights('/content/model_2_save2/unet_m2_best_model_e50+49.h5')
```

In [ ]:

```
!wget --header="Host: doc-10-3o-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" --heade
r="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="Cooki
e: AUTH_82jcsesreiehjbhkrct3c4mrj1raokod_nonce=nbiprhol3touc" --header="Connection: keep-alive" "https:
//doc-10-3o-docs.googleusercontent.com/docs/securesc/rg90kivf62vcrm9d2s7vb24hsj0c3fo2/h9jlc8qjfic9o71tc
7l53b16fbb7pg9h/1646372025000/16522560826923149764/00176583124175523585/1JZ8M4EiQvHAyugt3qa3xUse9_XdTdI
D5?e=download&ax=ACxEAsZ834z3IKgN4qDHUGebvZYBUb1__I2LZwwkqLujCJJgKvztbdoDoU32BSrjLOmWGXFYGOKgdluvsEfV8X
thBCHztL7P3kBAnS4v9yNgdWmmjVMP4KgQhVrI9NQmPKLZZh7K2SLIni12NEz_RSsjWTP9VoqNt0pmZAr9c_zwt9oeWPuoLf78j-vWB
JMK3E0CNM8E3MPnu7nrAWQ3B3mb2a6HOzD6V97YFwSY_j7E5jpikSaQwtAnv1B4rpnGOwErtcWvJ5Z5Mad1MeV3yxdst7vqK7tz5oL1
OB7qSf9Bd9gx7mCuzBB-tXSHrZ7eHOkvUyTydqC-JKYz_QzZ6zTSEfXm4YJNZiMmmTBCn2wg5cssMmuc-YtRt-UvLnS3vzlKt6pxDx0
vi3Aizql9HcV59Ni4Bll_ToXa2zG4iQSlvwXCTktRd8kEFoYQfWnqyAipresTepnwev9ImIpmlvqvyjc6A5jtiwjul6mfJiDffLjst5
3xEgGfVlmIQ3v1iAdfV-oa_un7JytYjqWaDx0hnsk1BHzT0s5aGKofiBjoP6aHS-EfV51dI9I41Tgm4WrJIe3RCksNQEzdghzHb7-o3
Huyx83-KqSfkLICjsQSLxHJoqxWmMRf9wW9Kwrkyt_HDXD1KSCm3J364EzY7dXEENN0W4NUj3R-DVK_fURH&authuser=0&nonce=nb
iprhol3touc&user=00176583124175523585&hash=9af4sf51kra0lbh2lktlff0oae5n2neq" -c -O 'unet_m2_best_model_
e50+49.h5'
```

```
--2022-03-04 05:34:54--  https://doc-10-3o-docs.googleusercontent.com/docs/securesc/rg90kivf62vcrm9d2s7
vb24hsj0c3fo2/h9jlc8qjfic9o71tc7l53b16fbb7pg9h/1646372025000/16522560826923149764/00176583124175523585/
1JZ8M4EiQvHAyugt3qa3xUse9_XdTdID5?e=download&ax=ACxEAsZ834z3IKgN4qDHUGebvZYBUb1__I2LZwwkqLujCJJgKvztbdo
DoU32BSrjLOmWGXFYGOKgdluvsEfV8XthBCHztL7P3kBAnS4v9yNgdWmmjVMP4KgQhVrI9NQmPKLZZh7K2SLIni12NEz_RSsjWTP9Vo
qNt0pmZAr9c_zwt9oeWPuoLf78j-vWBJMK3E0CNM8E3MPnu7nrAWQ3B3mb2a6HOzD6V97YFwSY_j7E5jpikSaQwtAnv1B4rpnGOwErt
cWvJ5Z5Mad1MeV3yxdst7vqK7tz5oL1OB7qSf9Bd9gx7mCuzBB-tXSHrZ7eHOkvUyTydqC-JKYz_QzZ6zTSEfXm4YJNZiMmmTBCn2wg
5cssMmuc-YtRt-UvLnS3vzlKt6pxDx0vi3Aizql9HcV59Ni4Bll_ToXa2zG4iQSlvwXCTktRd8kEFoYQfWnqyAipresTepnwev9ImIp
mlvqvyjc6A5jtiwjul6mfJiDffLjst53xEgGfVlmIQ3v1iAdfV-oa_un7JytYjqWaDx0hnsk1BHzT0s5aGKofiBjoP6aHS-EfV51dI9
I41Tgm4WrJIe3RCksNQEzdghzHb7-o3Huyx83-KqSfkLICjsQSLxHJoqxWmMRf9wW9Kwrkyt_HDXD1KSCm3J364EzY7dXEENN0W4NUj
3R-DVK_fURH&authuser=0&nonce=nbiprhol3touc&user=00176583124175523585&hash=9af4sf51kra0lbh2lktlff0oae5n2
neq
```

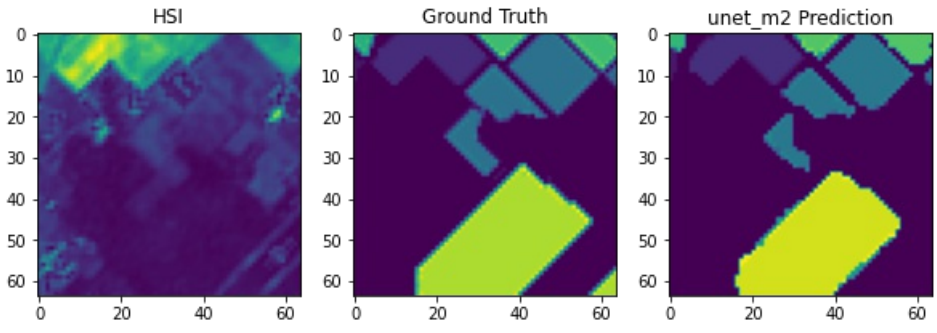
```
Resolving doc-10-3o-docs.googleusercontent.com (doc-10-3o-docs.googleusercontent.com)... 142.251.6.132,
2607:f8b0:4001:c5a::84
Connecting to doc-10-3o-docs.googleusercontent.com (doc-10-3o-docs.googleusercontent.com)|142.251.6.132
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28038648 (27M) [application/octet-stream]
Saving to: 'unet_m2_best_model_e50+49.h5'

unet_m2_best_model_ 100%[===================>]  26.74M  52.2MB/s    in 0.5s

2022-03-04 05:34:55 (52.2 MB/s) - 'unet_m2_best_model_e50+49.h5' saved [28038648/28038648]
```

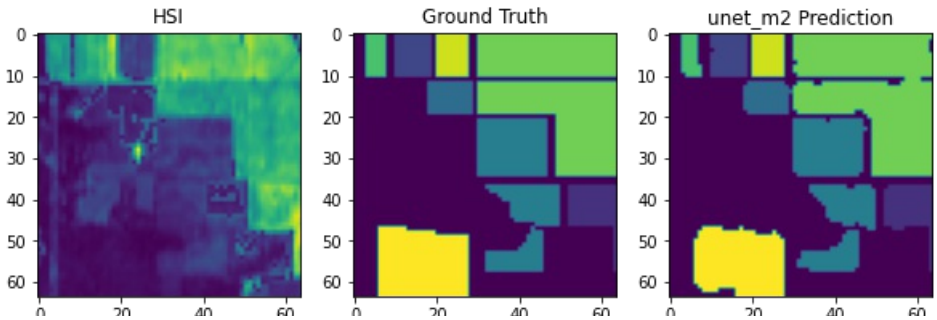
In [ ]:

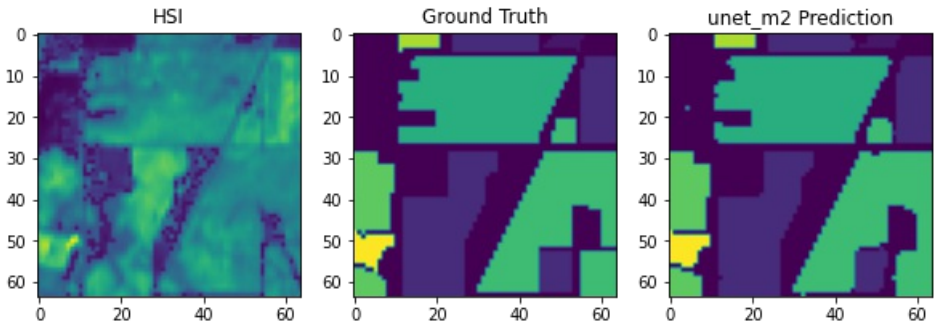
```
# Loading saved model weights
unet_m2.load_weights('unet_m2_best_model_e50+49.h5')
```

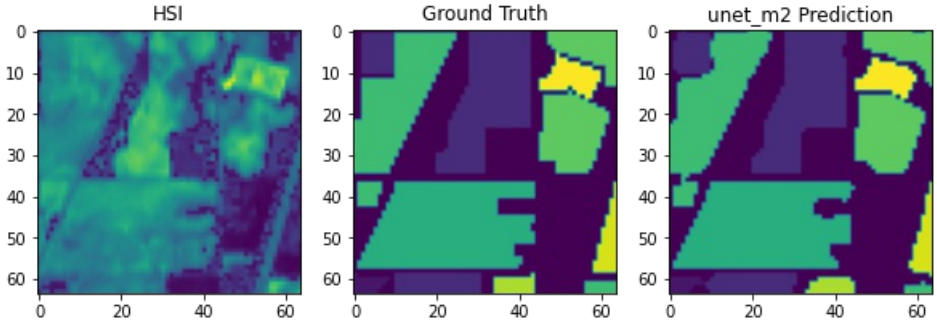
In [ ]:

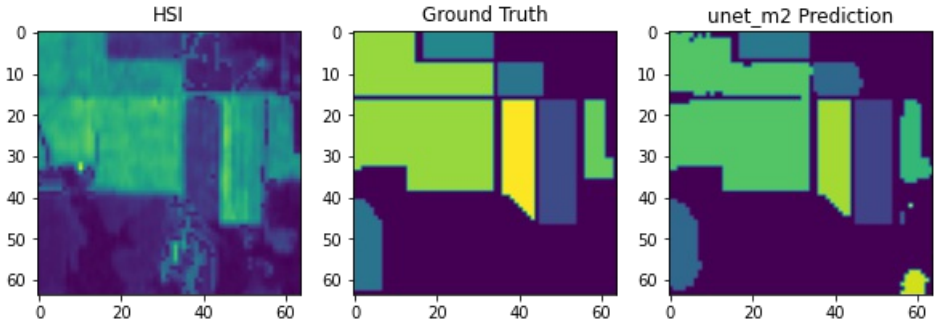
```
# Plotting Model prediction of segmentation alongside HSI and Ground Truth
i=0
for im, gt in zip(X_test[20:100],y_test[20:100]):

    # 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
```

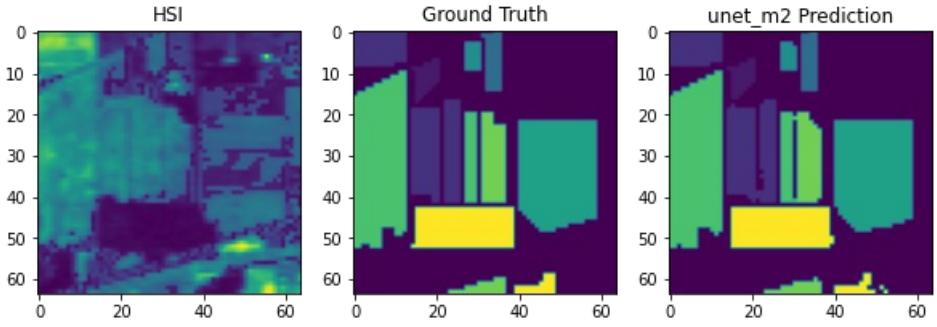


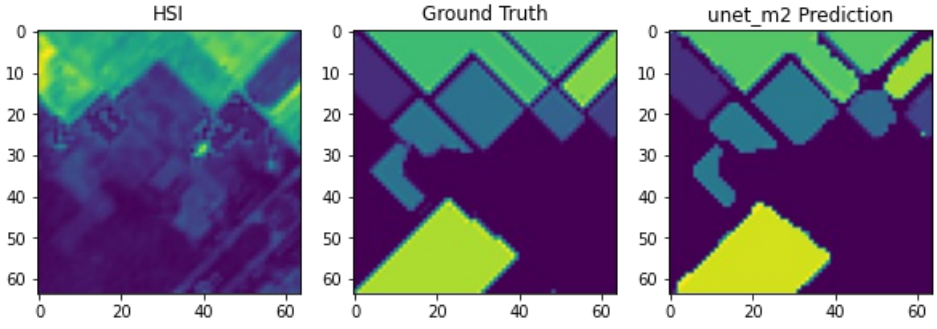



HSI
Ground Truth
unet_m2 Prediction

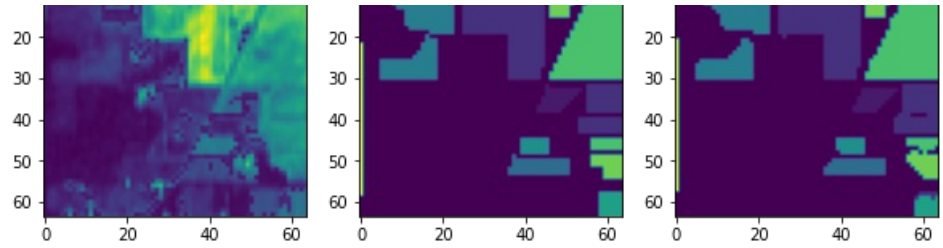

HSI
Ground Truth
unet_m2 Prediction

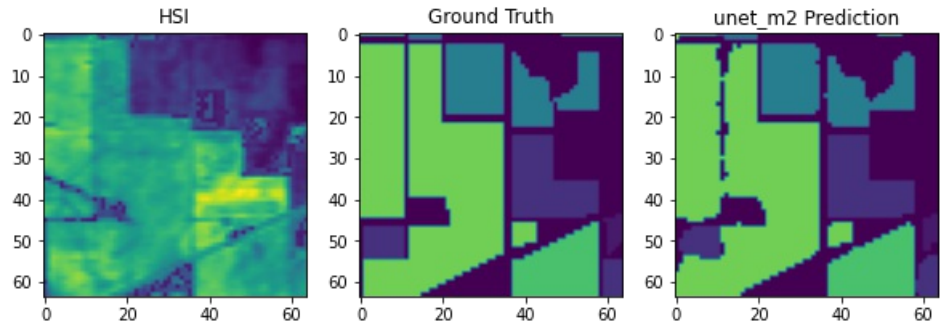

HSI
Ground Truth
unet_m2 Prediction

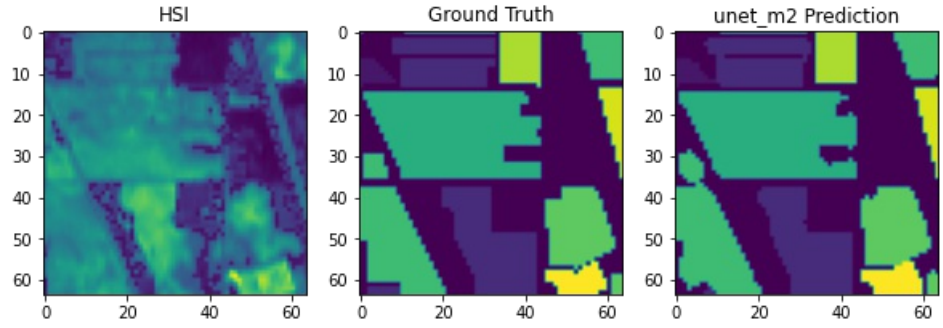

HSI
Ground Truth
unet_m2 Prediction

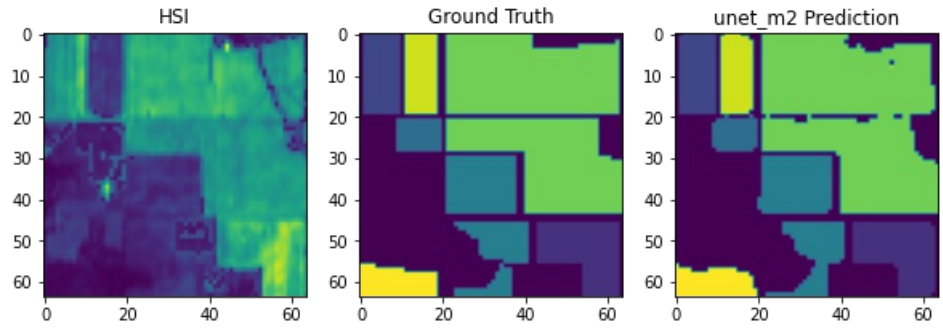

HSI
Ground Truth
unet_m2 Prediction


HSI
Ground Truth
unet_m2 Prediction









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
```

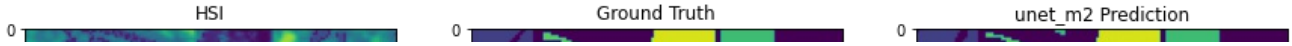
Unpatchified prediction

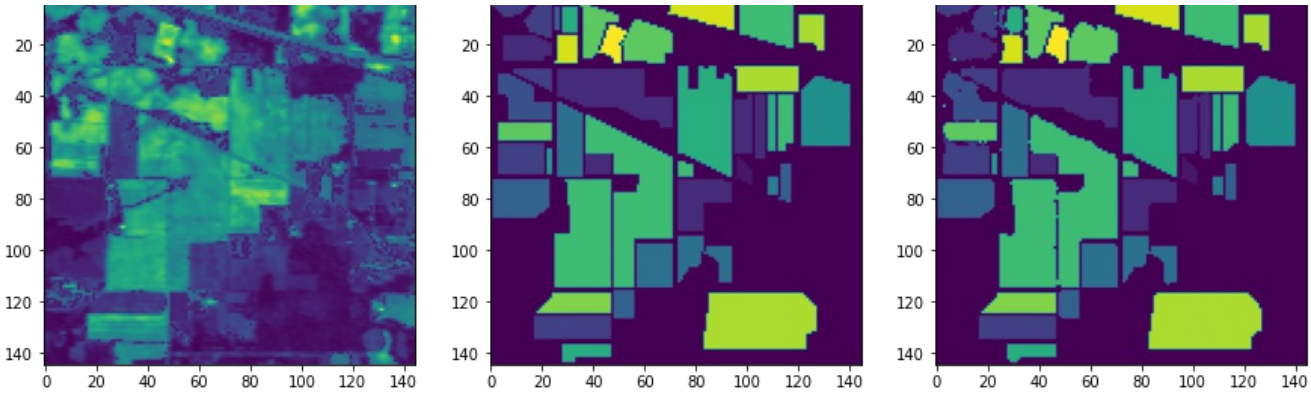
In [ ]:

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

In [ ]:

```
# plottng 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 probablities(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 [ ]:

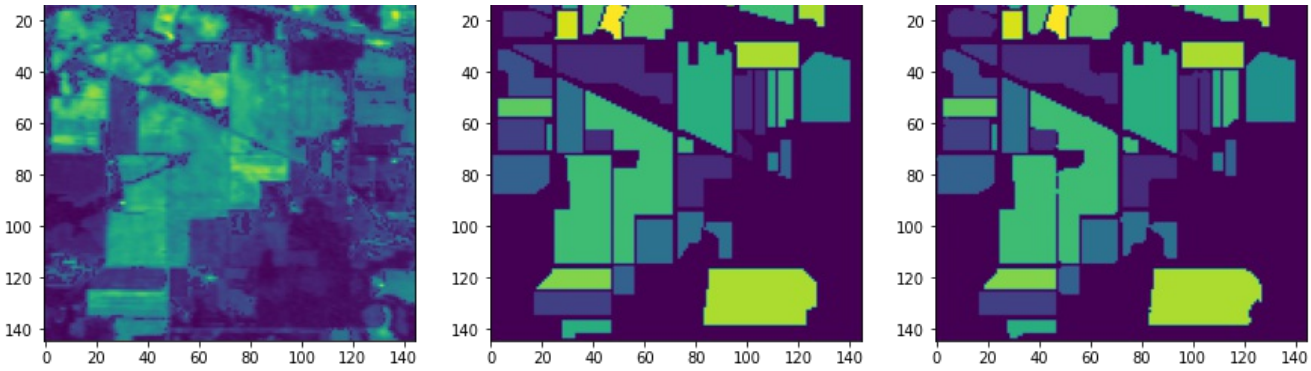
```
# Generating the 3D probabilities grid of all patches associated with full image.
grid = 0
grp = 0
img_prediction = np.zeros((145, 145, 100*17))
for i in range(10):
  for j in range(10):
    img_prediction[i*9:i*9+64,
                   j*9:j*9+64,
                   grp:grp+17] = pred[grid]
   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

```
from sklearn.metrics import f1_score, cohen_kappa_score
```

In [ ]:

```
F1_unet_m1 = f1_score(y,y_hat,average='micro')
print('micro F1 score of simple unet model for full image : ',F1_unet_m1)
kappa_unet_m1 = cohen_kappa_score(y,y_hat)
print('kappa score of simple unet model for full image : ',kappa_unet_m1)
```

```
micro F1 score of simple unet model for full image :  0.9925326991676575
kappa score of simple unet model for full image :  0.9894364597996672
```

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()
```

In [ ]:

```
F1_unet_m1_val = f1_score(y_val,y_hat_val,average='micro')
print('micro F1 score of simple unet model for validation data: ',F1_unet_m1_val)
kappa_unet_m1_val = cohen_kappa_score(y_val,y_hat_val)
print('kappa score of simple unet model for validation data: ',kappa_unet_m1_val)
```

```
micro F1 score of simple unet model for validation data:  0.95829345703125
kappa score of simple unet model for validation data:  0.9454368938453068
```

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()
```

Observations:

1. Pretrained U-Net

Model was trained for 50 epochs

- Scores for Full image prediction (Train and test data combined):
  - micro F1 score : 86.6%
  - kappa score : 80.29%
- Scores for test image prediction (only test data/validation data):
  - micro F1 score : 72.68%
  - kappa score : 63.15%
- Though the scores are better for full image, the segmented images are more like globules. This might be due to the model which are trained weights(imagenet) which are trained specifically for RGB images.

---

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 test data combined):
  - micro F1 score : 99.25%
  - kappa score : 98.94%
- Scores for test image prediction (only test data/validation data):
  - micro F1 score : 95.82%
  - kappa score : 94.54%
- 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

For Classifying the HS Image of broader classes, Simple U-Net model can be considered and trained from scartch.

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