Deep Learning Project

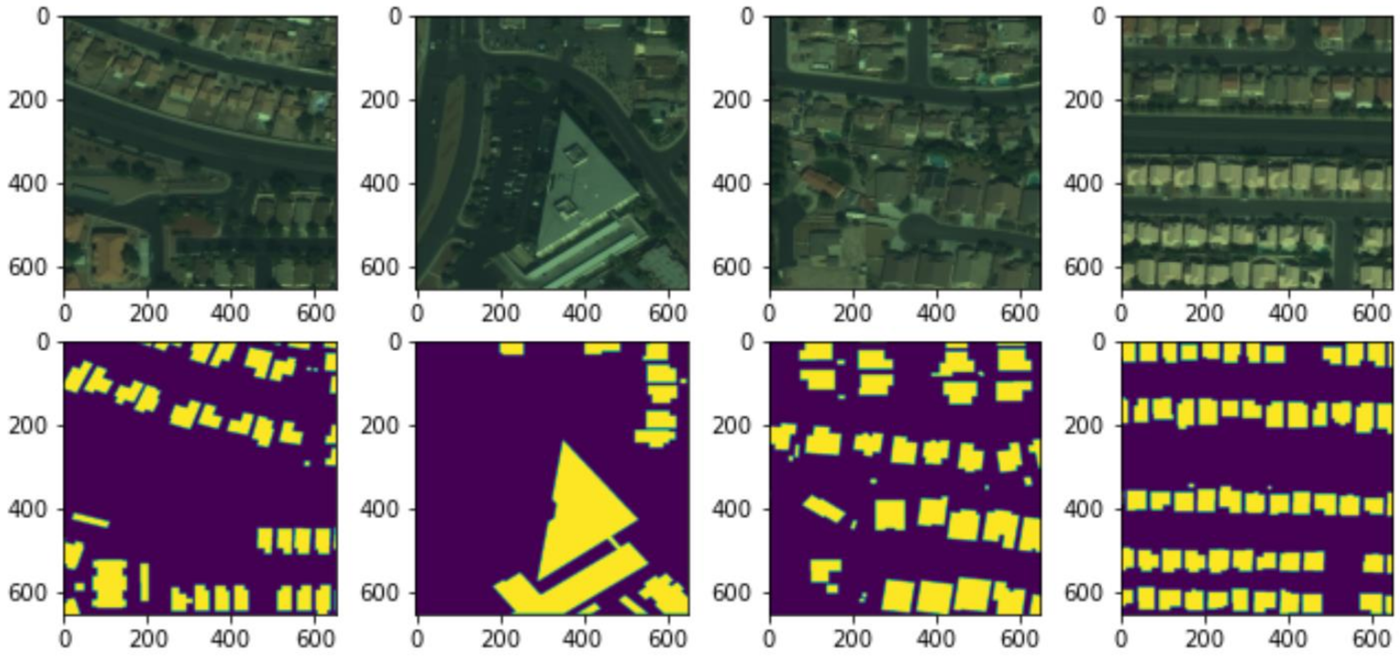
Harold Gamarro

Joe Brian Malubay

Said Mejia

Juan Pablo Montoya

Spacenet Data processing for building detection



Input Image:

- 3 band geotiff
- 650 x 650 x 3
- ~ 3000 images

Input mask:

- Geojson
- 650 x 650 x 3

Image and mask generator

Output data:

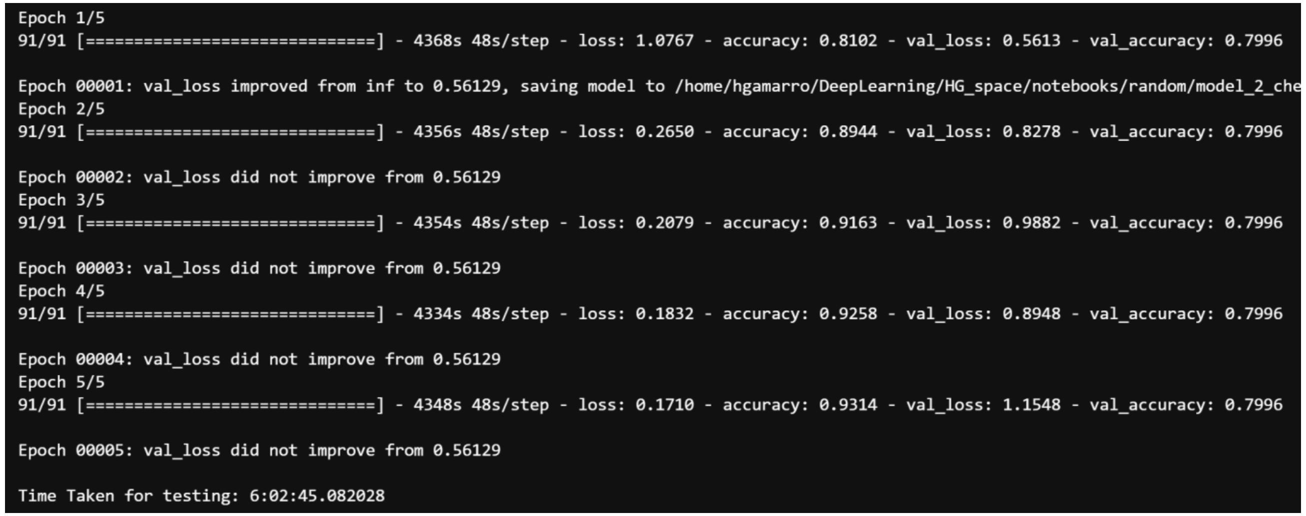
- Processed NumPy arrays in batches of 32 images
- Normalized data
- Rescaled to 512 x 512

- 91 files of 32x512x512x3 for images
- 91 files of 32x512x512x1 for imagmaskses

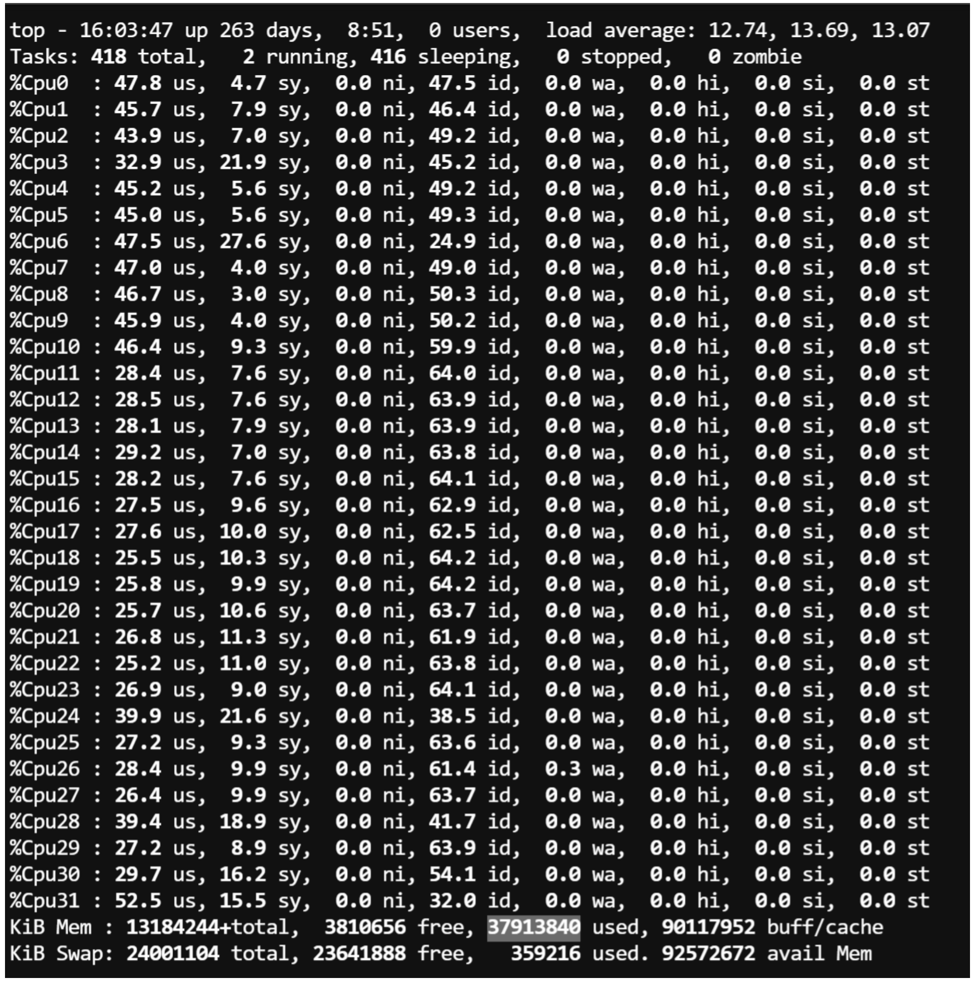
Spacenet Data processing for building detection

Current Progress:

- Finished source code to convert images to jpegs for model input
- Tested out the framework to load image and mask into a Unet model that successfully trained model over 5 epochs
- Computation served as a big challenge resulting in OOM errors
    - Figured out how to use research compute cluster to handle the training
    - 32 cores with about 60 GB of memory
    - Can submit jobs and test various models

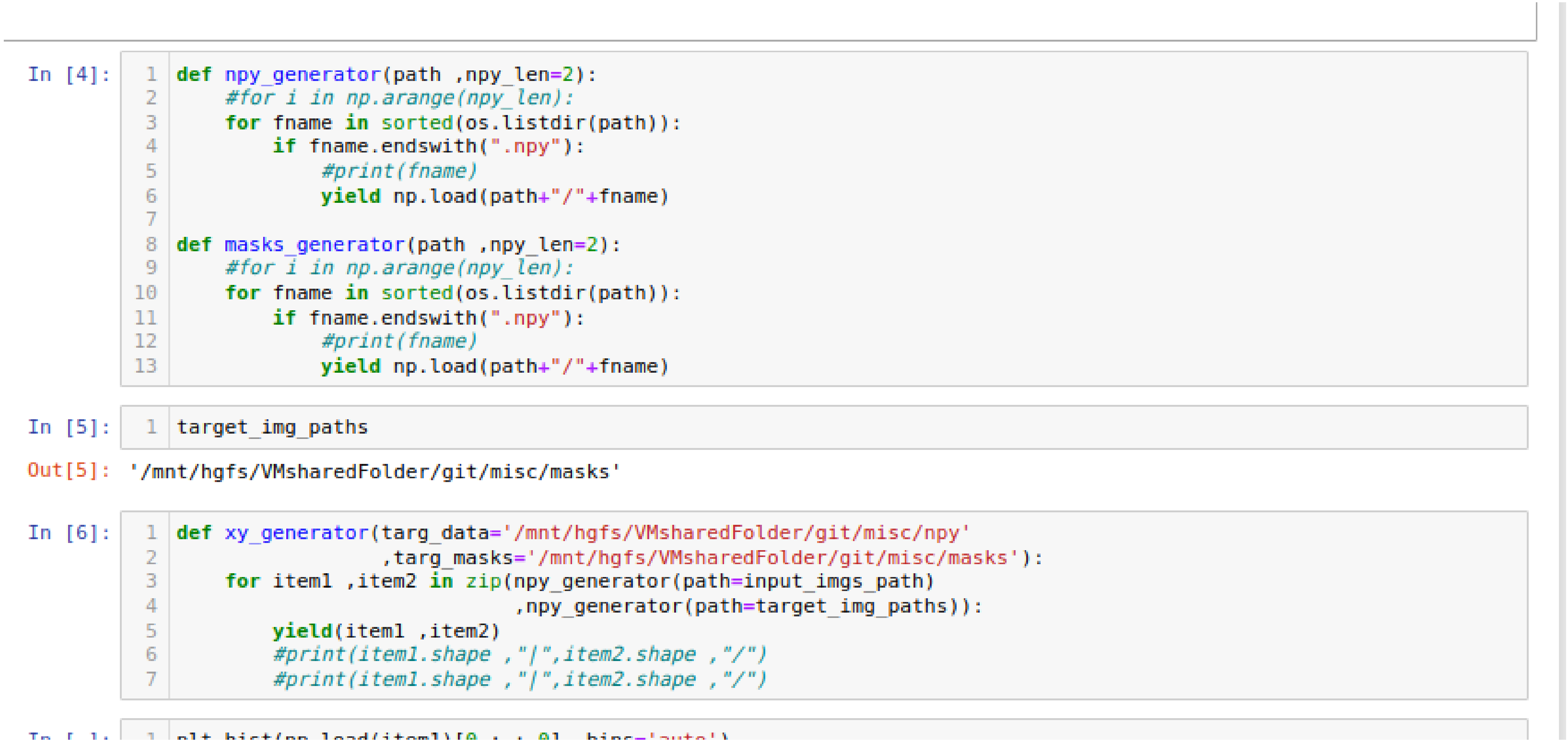


```
Epoch 1/5
91/91 [=============================] - 4368s 48s/step - loss: 1.0767 - accuracy: 0.8102 - val_loss: 0.5613 - val_accuracy: 0.7996

Epoch 00001: val_loss improved from inf to 0.56129, saving model to /home/hgamarro/DeepLearning/HG_space/notebooks/random/model_2_che
Epoch 2/5
91/91 [=============================] - 4356s 48s/step - loss: 0.2650 - accuracy: 0.8944 - val_loss: 0.8278 - val_accuracy: 0.7996

Epoch 00002: val_loss did not improve from 0.56129
Epoch 3/5
91/91 [=============================] - 4354s 48s/step - loss: 0.2079 - accuracy: 0.9163 - val_loss: 0.9882 - val_accuracy: 0.7996

Epoch 00003: val_loss did not improve from 0.56129
Epoch 4/5
91/91 [=============================] - 4334s 48s/step - loss: 0.1832 - accuracy: 0.9258 - val_loss: 0.8948 - val_accuracy: 0.7996

Epoch 00004: val_loss did not improve from 0.56129
Epoch 5/5
91/91 [=============================] - 4348s 48s/step - loss: 0.1710 - accuracy: 0.9314 - val_loss: 1.1548 - val_accuracy: 0.7996

Epoch 00005: val_loss did not improve from 0.56129

Time Taken for testing: 6:02:45.082028
```



```
top - 16:03:47 up 263 days,  8:51,  0 users,  load average: 12.74, 13.69, 13.07
Tasks: 418 total,   2 running, 416 sleeping,   0 stopped,   0 zombie
%Cpu0  : 47.8 us,  4.7 sy,  0.0 ni, 47.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  : 45.7 us,  7.9 sy,  0.0 ni, 46.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  : 43.9 us,  7.0 sy,  0.0 ni, 49.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  : 32.9 us, 21.9 sy,  0.0 ni, 45.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  : 45.2 us,  5.6 sy,  0.0 ni, 49.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  : 45.0 us,  5.6 sy,  0.0 ni, 49.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  : 47.5 us, 27.6 sy,  0.0 ni, 24.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  : 47.0 us,  4.0 sy,  0.0 ni, 49.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu8  : 46.7 us,  3.0 sy,  0.0 ni, 50.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu9  : 45.9 us,  4.0 sy,  0.0 ni, 50.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu10 : 46.4 us,  9.3 sy,  0.0 ni, 59.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu11 : 28.4 us,  7.6 sy,  0.0 ni, 64.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu12 : 28.5 us,  7.6 sy,  0.0 ni, 63.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu13 : 28.1 us,  7.9 sy,  0.0 ni, 63.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu14 : 29.2 us,  7.0 sy,  0.0 ni, 63.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu15 : 28.2 us,  7.6 sy,  0.0 ni, 64.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu16 : 27.5 us,  9.6 sy,  0.0 ni, 62.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu17 : 27.6 us, 10.0 sy,  0.0 ni, 62.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu18 : 25.5 us, 10.3 sy,  0.0 ni, 64.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu19 : 25.8 us,  9.9 sy,  0.0 ni, 64.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu20 : 25.7 us, 10.6 sy,  0.0 ni, 63.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu21 : 26.8 us, 11.3 sy,  0.0 ni, 61.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu22 : 25.2 us, 11.0 sy,  0.0 ni, 63.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu23 : 26.9 us,  9.0 sy,  0.0 ni, 64.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu24 : 39.9 us, 21.6 sy,  0.0 ni, 38.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu25 : 27.2 us,  9.3 sy,  0.0 ni, 63.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu26 : 28.4 us,  9.9 sy,  0.0 ni, 61.4 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu27 : 26.4 us,  9.9 sy,  0.0 ni, 63.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu28 : 39.4 us, 18.9 sy,  0.0 ni, 41.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu29 : 27.2 us,  8.9 sy,  0.0 ni, 63.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu30 : 29.7 us, 16.2 sy,  0.0 ni, 54.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu31 : 52.5 us, 15.5 sy,  0.0 ni, 32.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 13184244+total,  3810656 free, 37913840 used, 90117952 buff/cache
KiB Swap: 24001104 total, 23641888 free,   359216 used. 92572672 avail Mem
```

- Each Epoch takes about 1 hour 20 min
- Uses all 32 cores
- Memory peaks 40 GB in above



In [4]:
```python
def npy_generator(path ,npy_len=2):
    #for i in np.arange(npy_len):
    for fname in sorted(os.listdir(path)):
        if fname.endswith(".npy"):
            #print(fname)
            yield np.load(path+"/"+fname)

def masks_generator(path ,npy_len=2):
    #for i in np.arange(npy_len):
    for fname in sorted(os.listdir(path)):
        if fname.endswith(".npy"):
            #print(fname)
            yield np.load(path+"/"+fname)
```

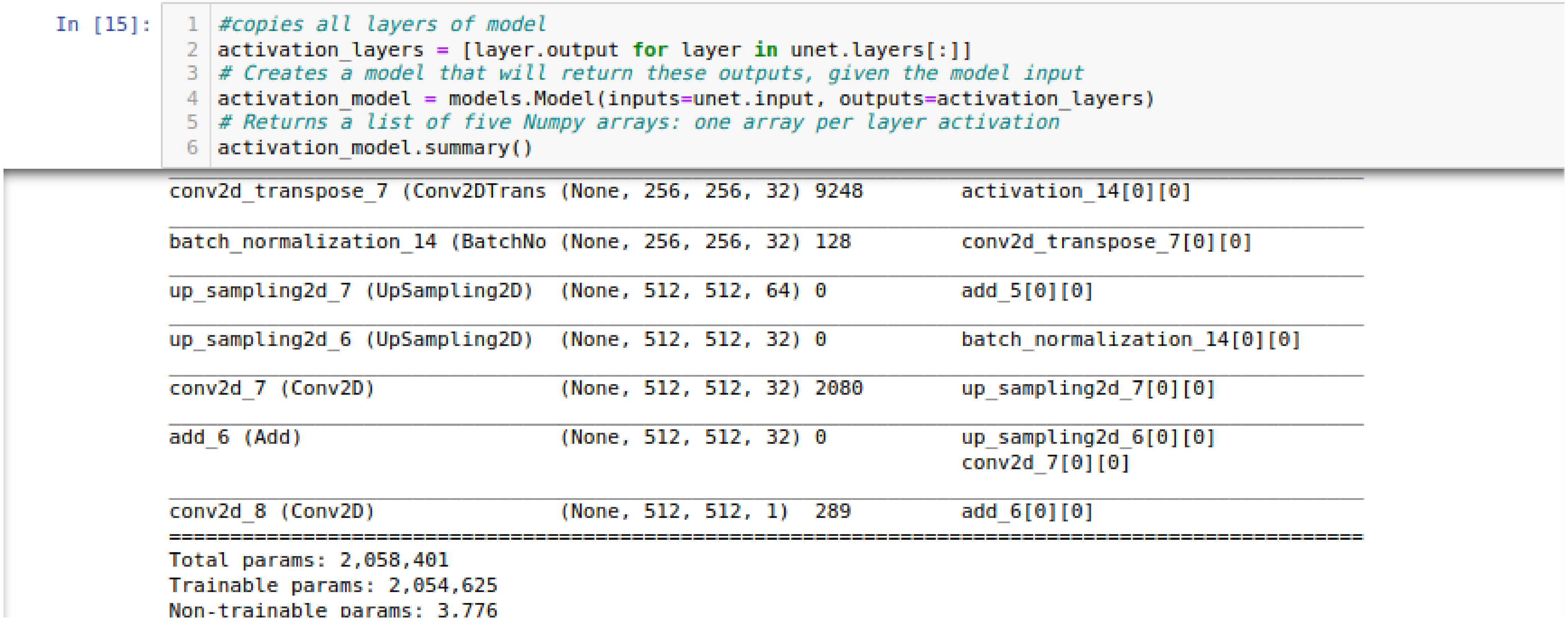
In [5]:
```python
target_img_paths
```

Out[5]:
```
'/mnt/hgfs/VMsharedFolder/git/misc/masks'
```

In [6]:
```python
def xy_generator(targ_data='/mnt/hgfs/VMsharedFolder/git/misc/npy'
                 ,targ_masks='/mnt/hgfs/VMsharedFolder/git/misc/masks'):
    for item1 ,item2 in zip(npy_generator(path=input_imgs_path)
                           ,npy_generator(path=target_img_paths)):
        yield(item1 ,item2)
        #print(item1.shape ,"|",item2.shape ,"/")
        #print(item1.shape ,"|",item2.shape ,"/")
```

In [ ]:
```python
plt.hist(np.load(item1)[0 : : 0] , bins='auto')
```

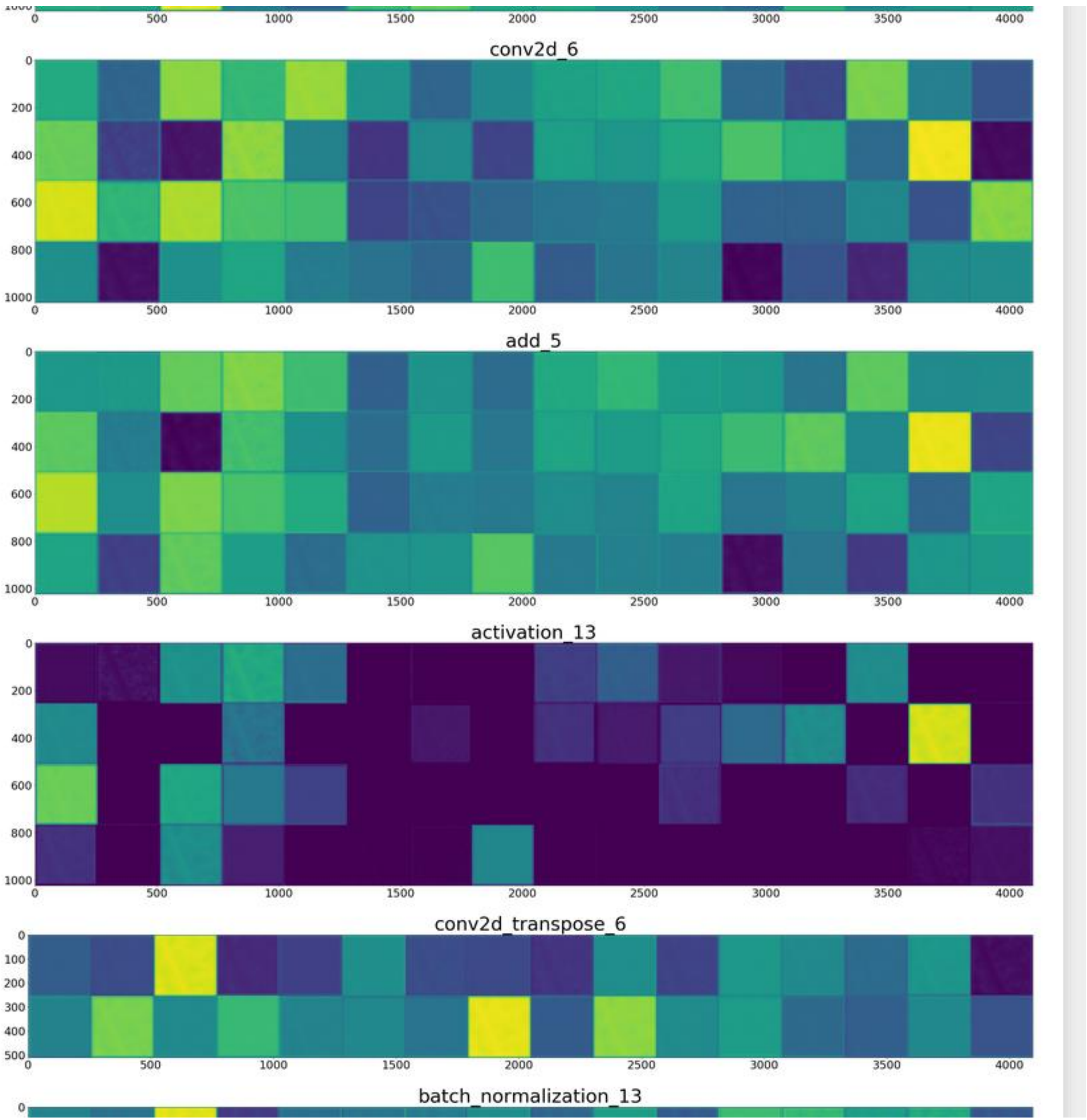
save_model_weights_hdf5(model, weight_path, overwrite = TRUE)

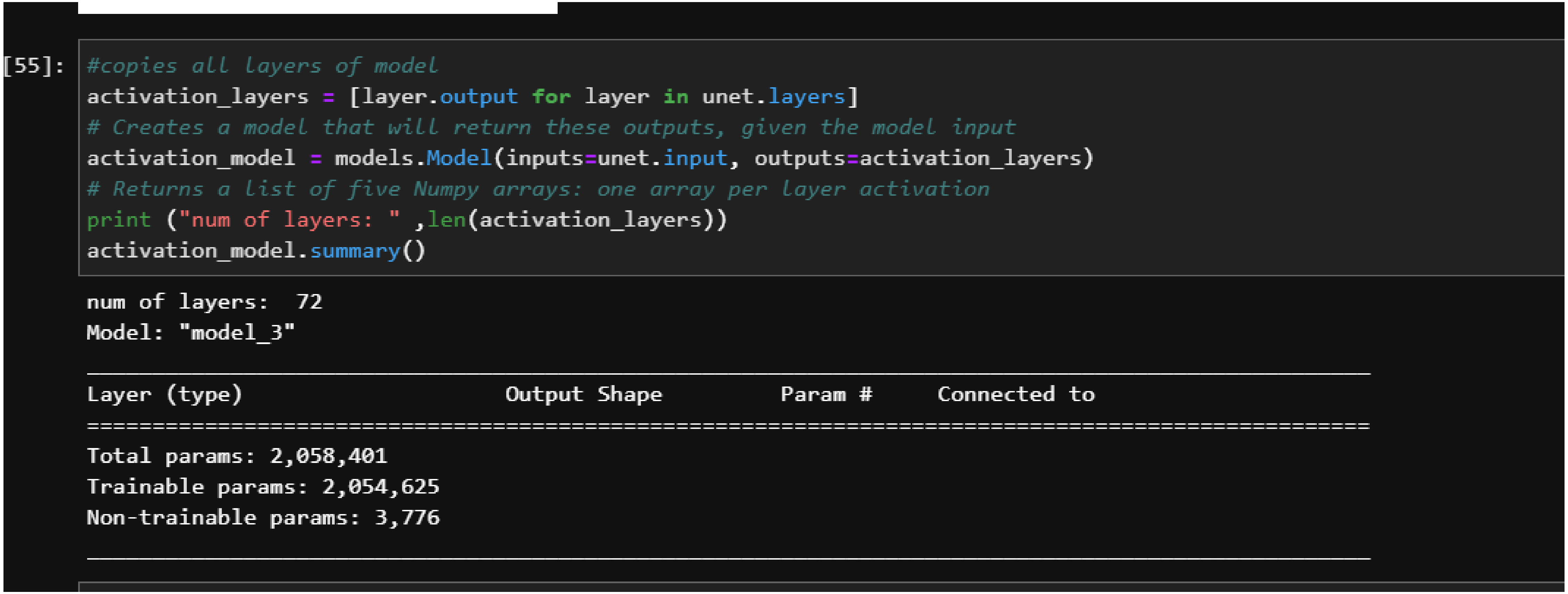


In [15]:

```
#copies all layers of model
activation_layers = [layer.output for layer in unet.layers[:]]
# Creates a model that will return these outputs, given the model input
activation_model = models.Model(inputs=unet.input, outputs=activation_layers)
# Returns a list of five Numpy arrays: one array per layer activation
activation_model.summary()
```

```
__________________________________________________________________________________________________
conv2d_transpose_7 (Conv2DTrans (None, 256, 256, 32) 9248        activation_14[0][0]
__________________________________________________________________________________________________
batch_normalization_14 (BatchNo (None, 256, 256, 32) 128         conv2d_transpose_7[0][0]
__________________________________________________________________________________________________
up_sampling2d_7 (UpSampling2D)  (None, 512, 512, 64) 0           add_5[0][0]
__________________________________________________________________________________________________
up_sampling2d_6 (UpSampling2D)  (None, 512, 512, 32) 0           batch_normalization_14[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 512, 512, 32) 2080        up_sampling2d_7[0][0]
__________________________________________________________________________________________________
add_6 (Add)                     (None, 512, 512, 32) 0           up_sampling2d_6[0][0]
                                                                 conv2d_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 512, 512, 1)  289         add_6[0][0]
==================================================================================================
Total params: 2,058,401
Trainable params: 2,054,625
Non-trainable params: 3,776
```

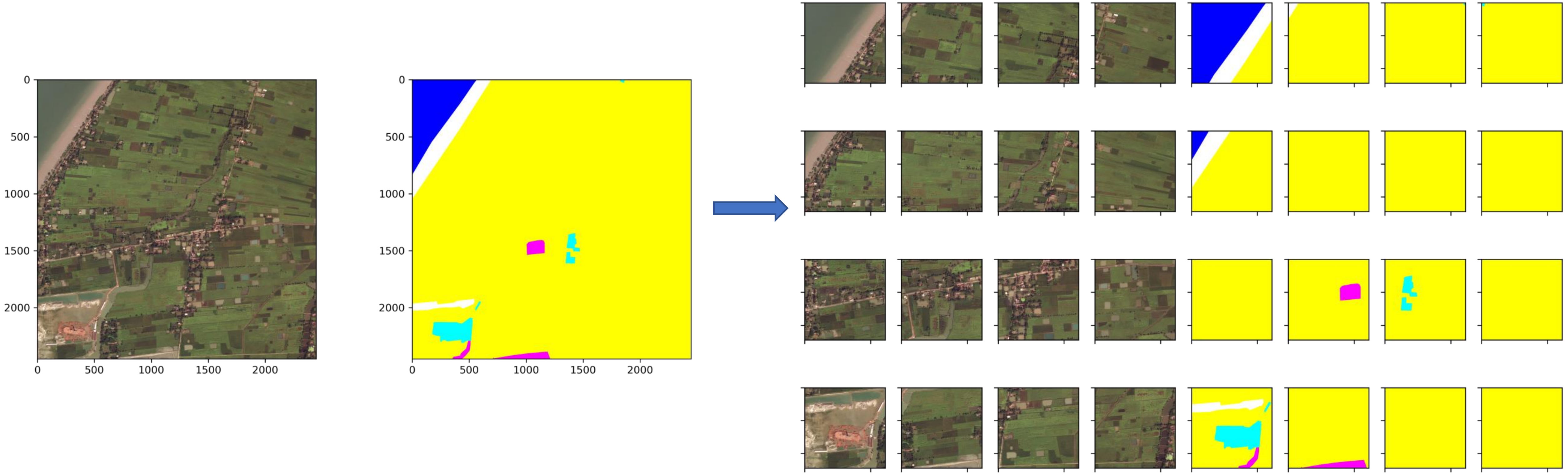

1000
0
500
1000
1500
2000
2500
3000
3500
4000
conv2d_6
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000
3500
4000
add_5
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000
3500
4000
activation_13
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000
3500
4000
conv2d_transpose_6
0
100
200
300
400
500
0
500
1000
1500
2000
2500
3000
3500
4000
batch_normalization_13
0



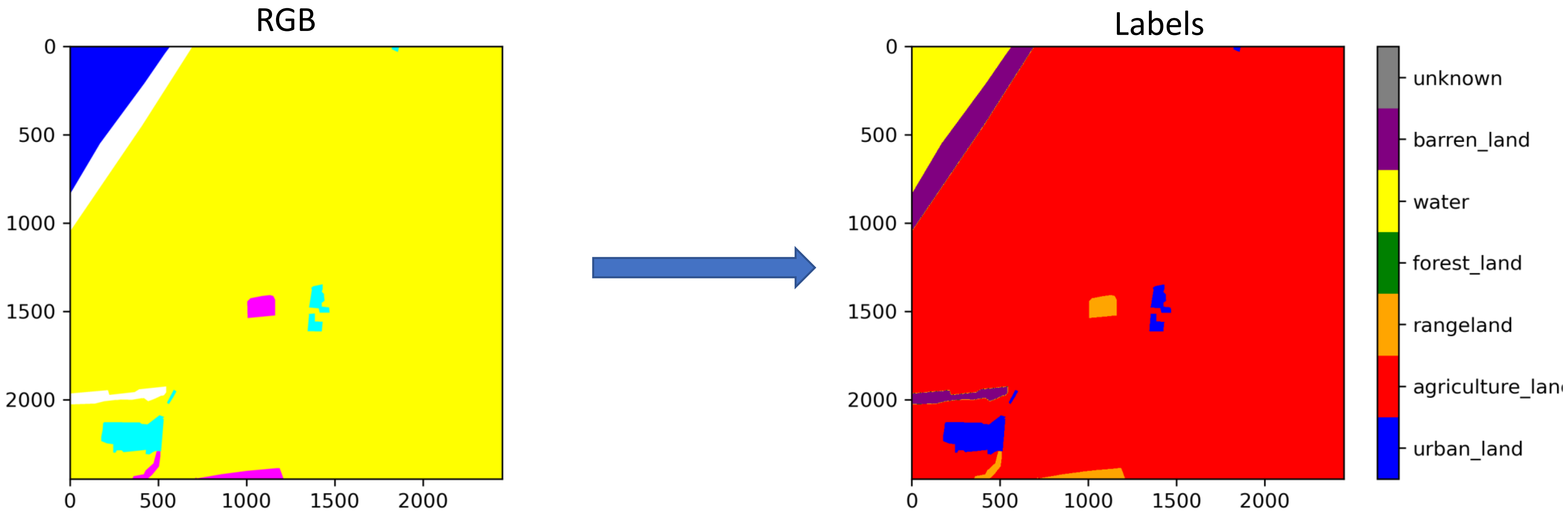
```
[55]: #copies all layers of model
      activation_layers = [layer.output for layer in unet.layers]
      # Creates a model that will return these outputs, given the model input
      activation_model = models.Model(inputs=unet.input, outputs=activation_layers)
      # Returns a list of five Numpy arrays: one array per layer activation
      print ("num of layers: " ,len(activation_layers))
      activation_model.summary()
```

```
num of layers:  72
Model: "model_3"

__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Total params: 2,058,401
Trainable params: 2,054,625
Non-trainable params: 3,776

__________________________________________________________________________________________________
```

Image Slicing



Convert Mask to Labels

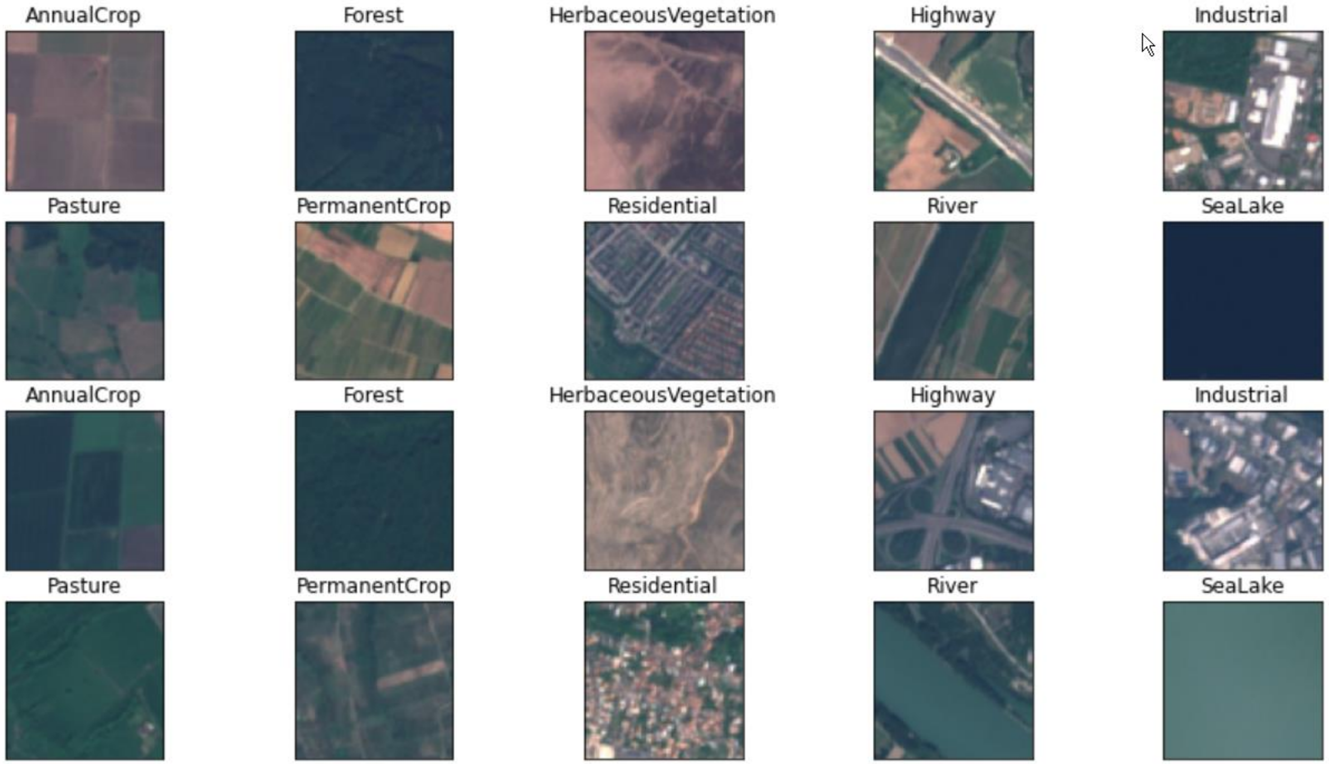


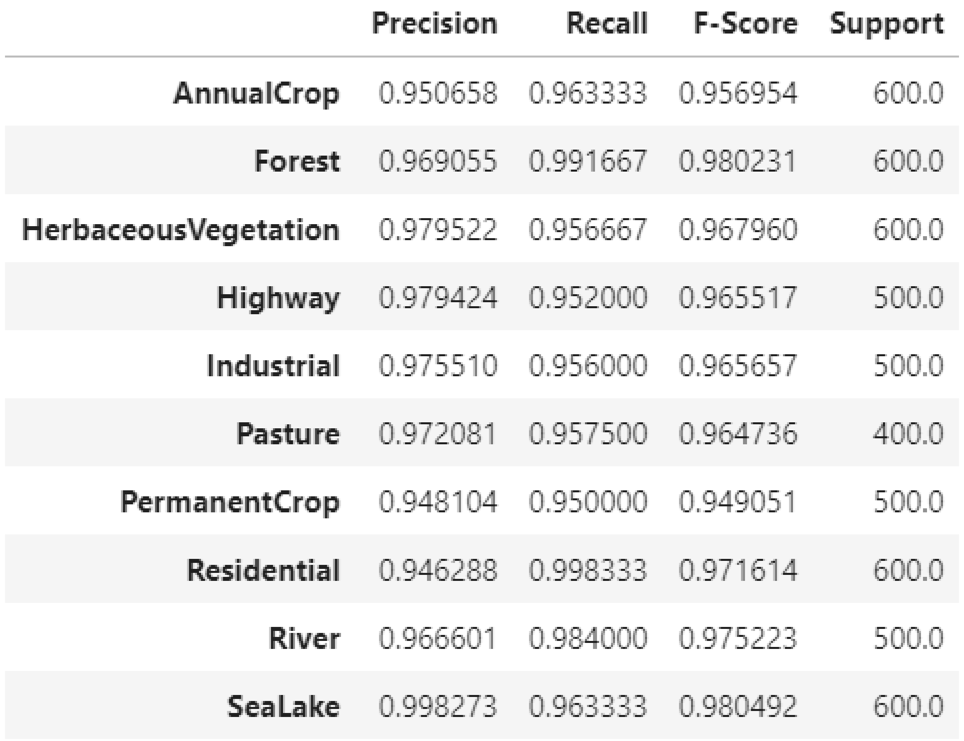
Processing Satellite image

- Convert from jp2 with 13 bands to png with just 3 bands (RBG)

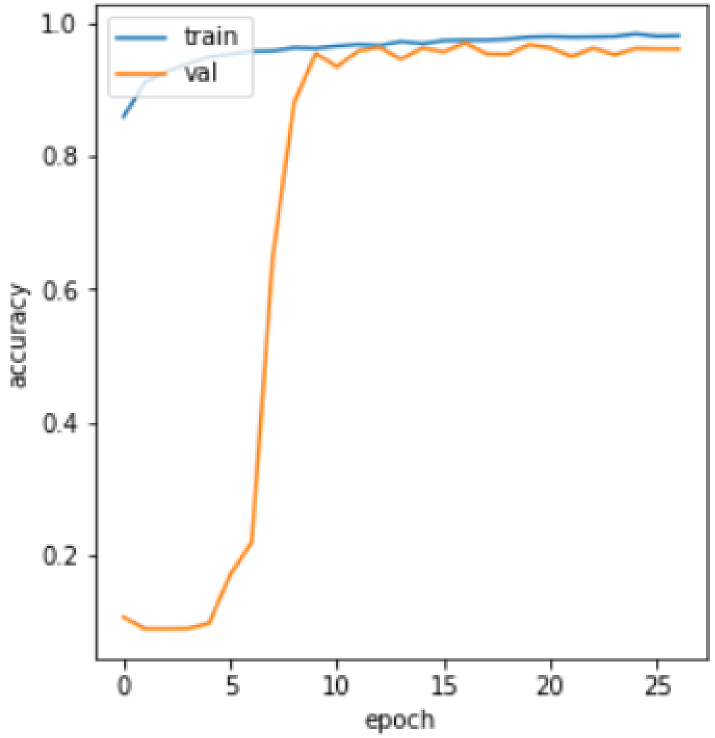
1st Dataset

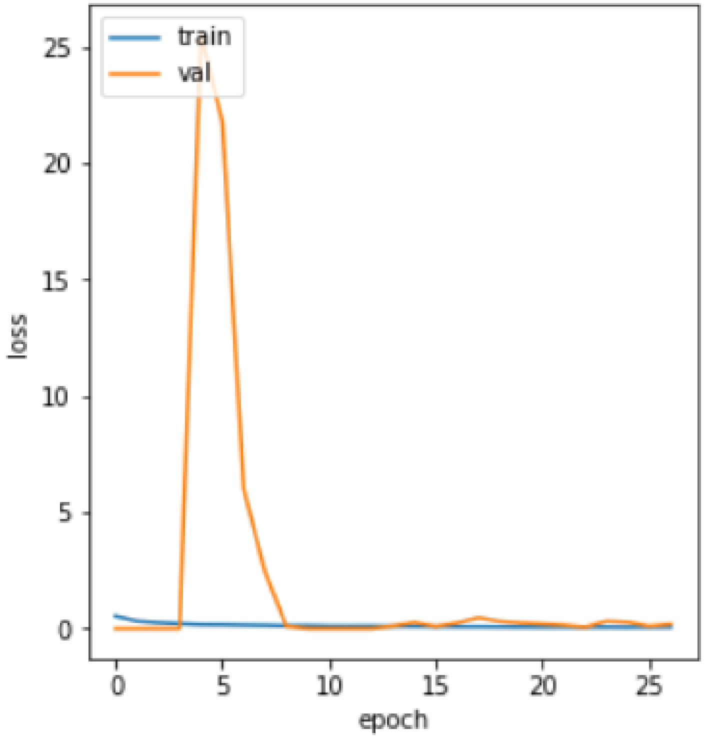
- Image (64x64x3)
- Using ResNet

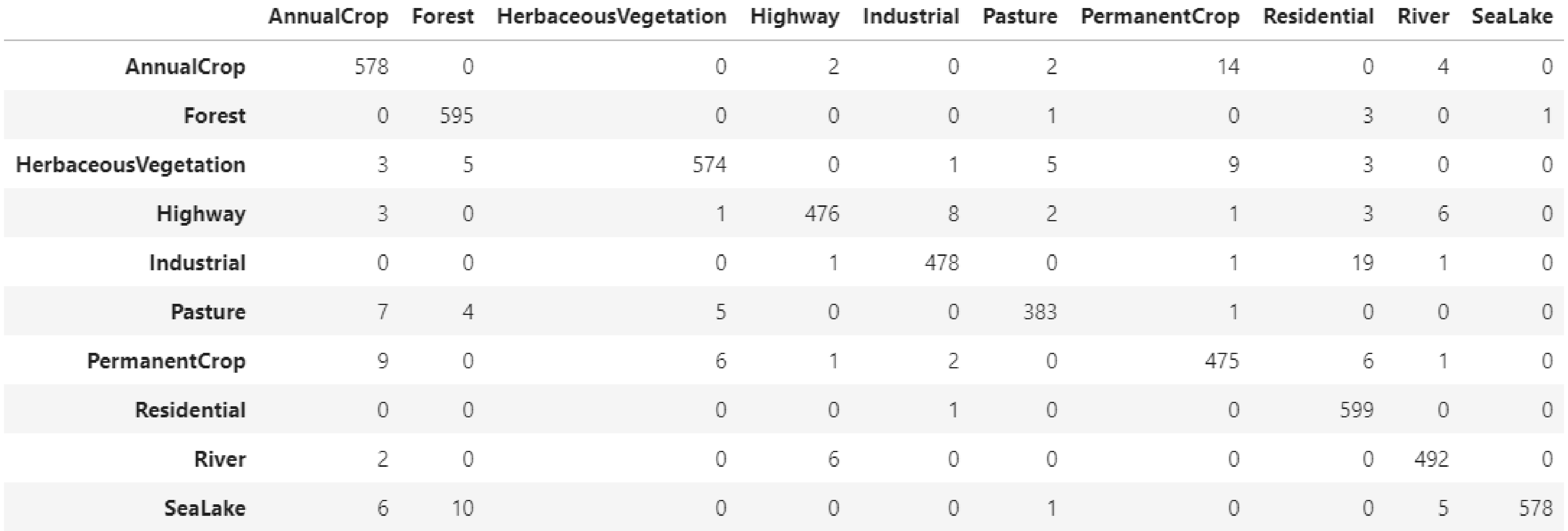




	Precision	Recall	F-Score	Support
AnnualCrop	0.950658	0.963333	0.956954	600.0
Forest	0.969055	0.991667	0.980231	600.0
HerbaceousVegetation	0.979522	0.956667	0.967960	600.0
Highway	0.979424	0.952000	0.965517	500.0
Industrial	0.975510	0.956000	0.965657	500.0
Pasture	0.972081	0.957500	0.964736	400.0
PermanentCrop	0.948104	0.950000	0.949051	500.0
Residential	0.946288	0.998333	0.971614	600.0
River	0.966601	0.984000	0.975223	500.0
SeaLake	0.998273	0.963333	0.980492	600.0



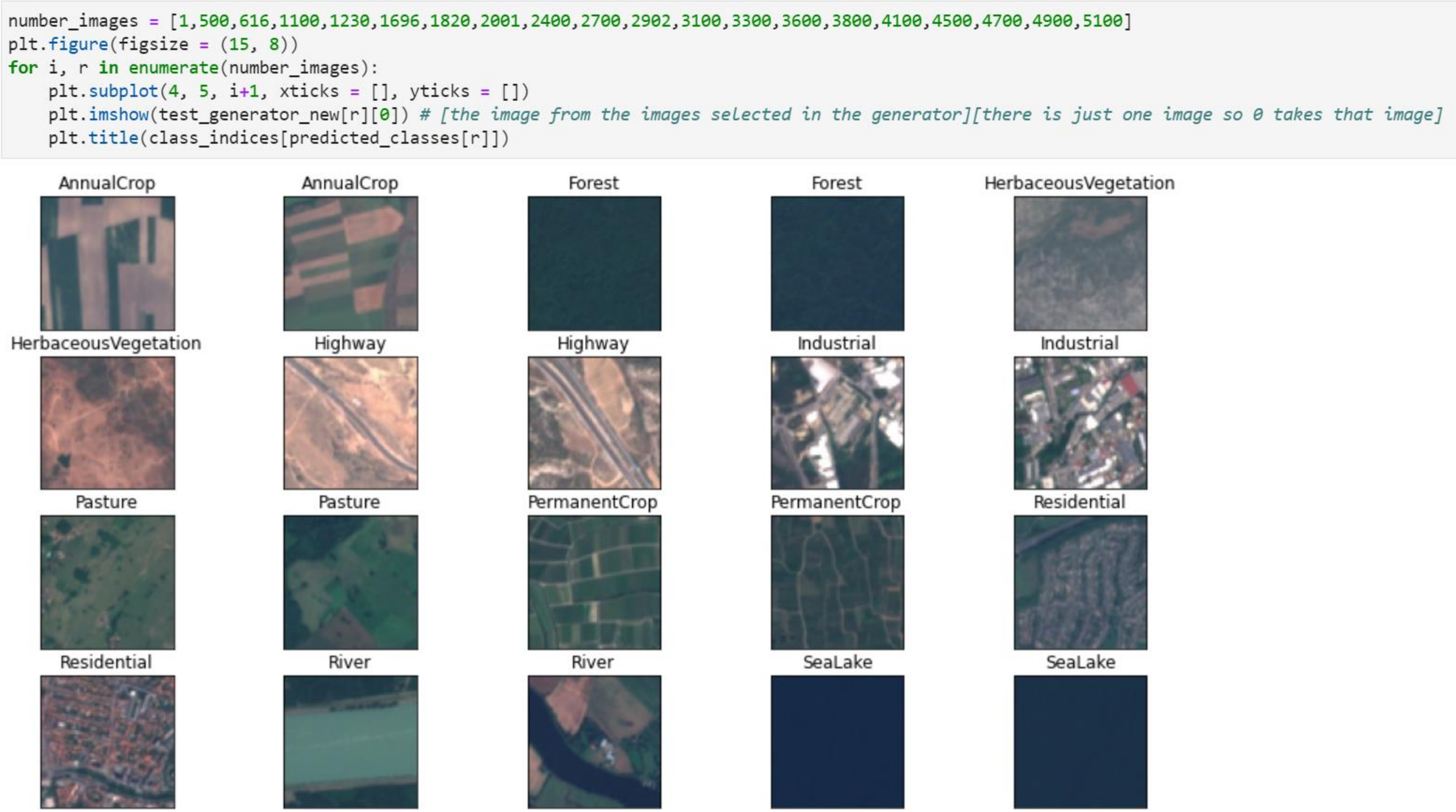




	AnnualCrop	Forest	HerbaceousVegetation	Highway	Industrial	Pasture	PermanentCrop	Residential	River	SeaLake
AnnualCrop	578	0	0	2	0	2	14	0	4	0
Forest	0	595	0	0	0	1	0	3	0	1
HerbaceousVegetation	3	5	574	0	1	5	9	3	0	0
Highway	3	0	1	476	8	2	1	3	6	0
Industrial	0	0	0	1	478	0	1	19	1	0
Pasture	7	4	5	0	0	383	1	0	0	0
PermanentCrop	9	0	6	1	2	0	475	6	1	0
Residential	0	0	0	0	1	0	0	599	0	0
River	2	0	0	6	0	0	0	0	492	0
SeaLake	6	10	0	0	0	1	0	0	5	578

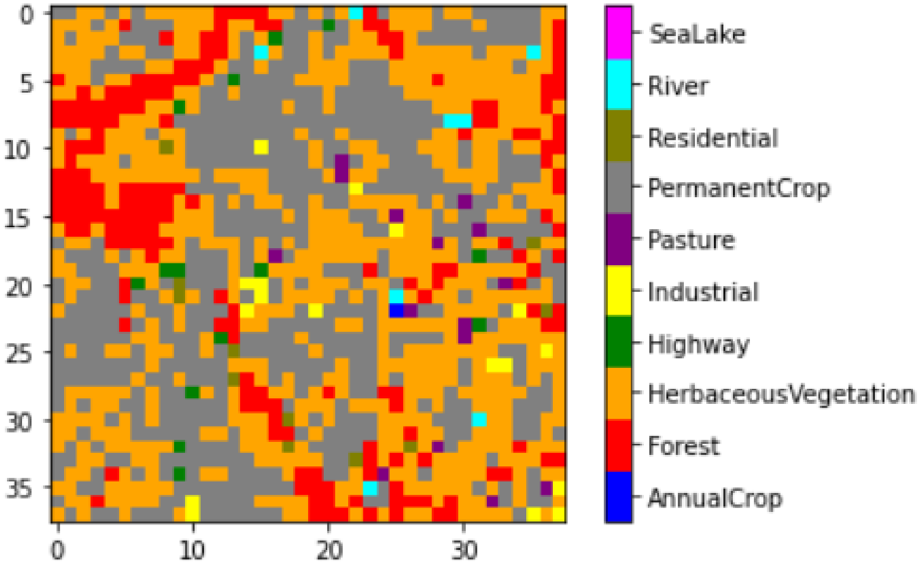
Results

- Image (64x64x3)
- Using ResNet



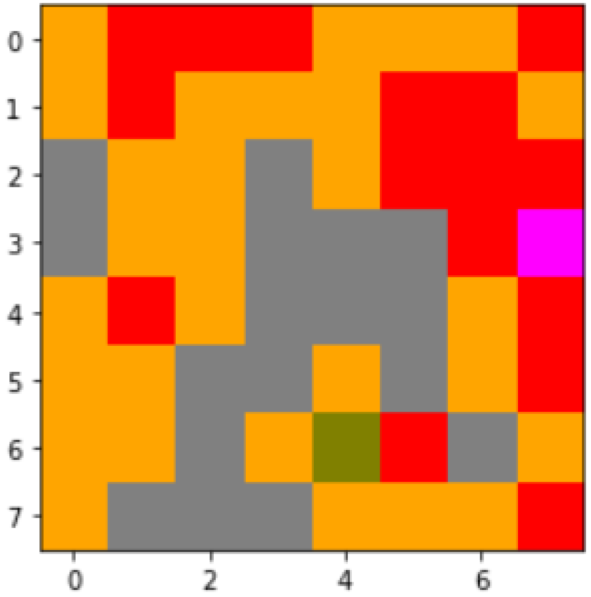
```
number_images = [1,500,616,1100,1230,1696,1820,2001,2400,2700,2902,3100,3300,3600,3800,4100,4500,4700,4900,5100]
plt.figure(figsize = (15, 8))
for i, r in enumerate(number_images):
    plt.subplot(4, 5, i+1, xticks = [], yticks = [])
    plt.imshow(test_generator_new[r][0]) # [the image from the images selected in the generator][there is just one image so 0 takes that image]
    plt.title(class_indices[predicted_classes[r]])
```

2448x2448x3

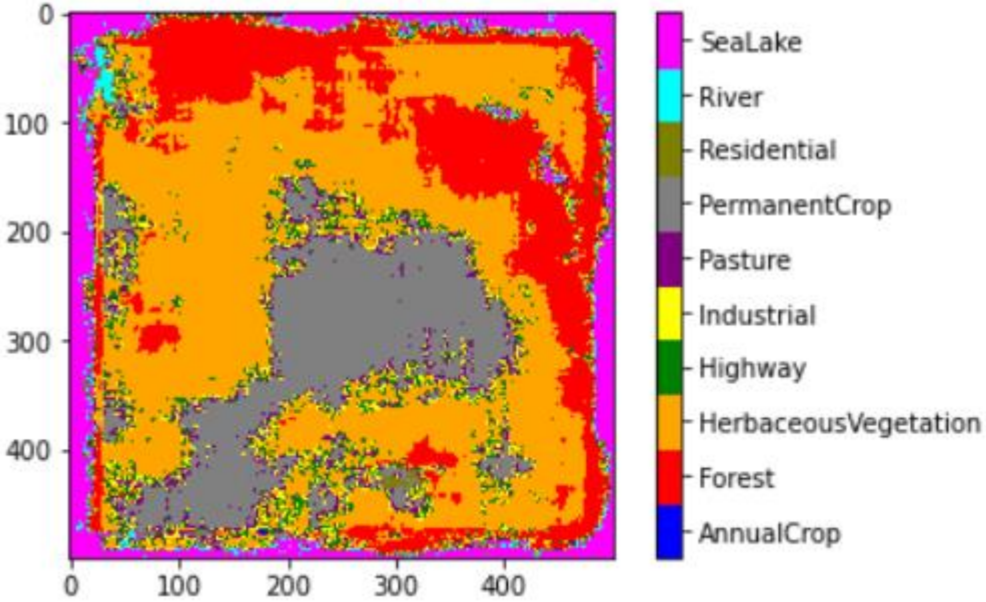


64x64

564x564x3



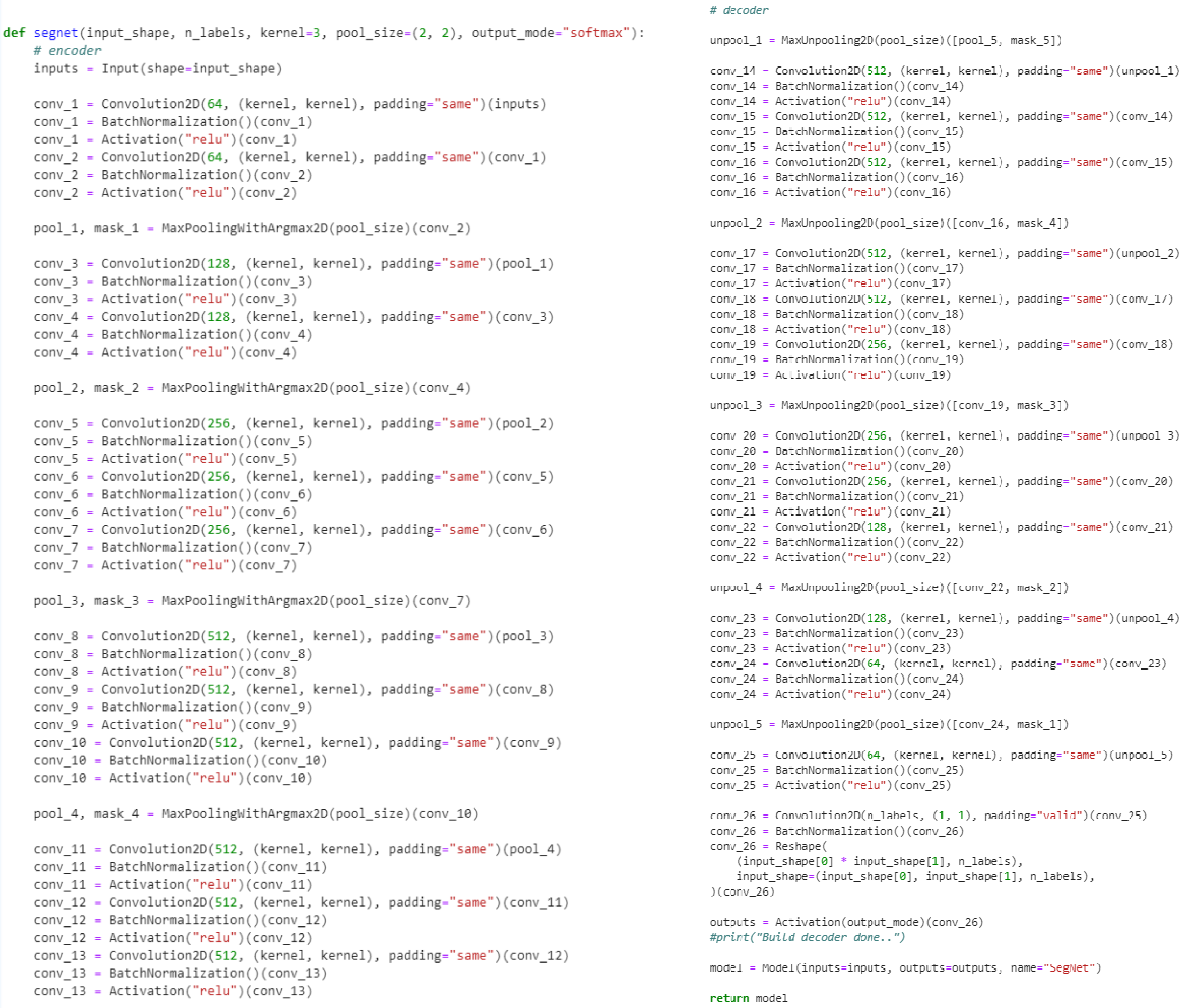
64x64



64x64 padded image

2nd Dataset

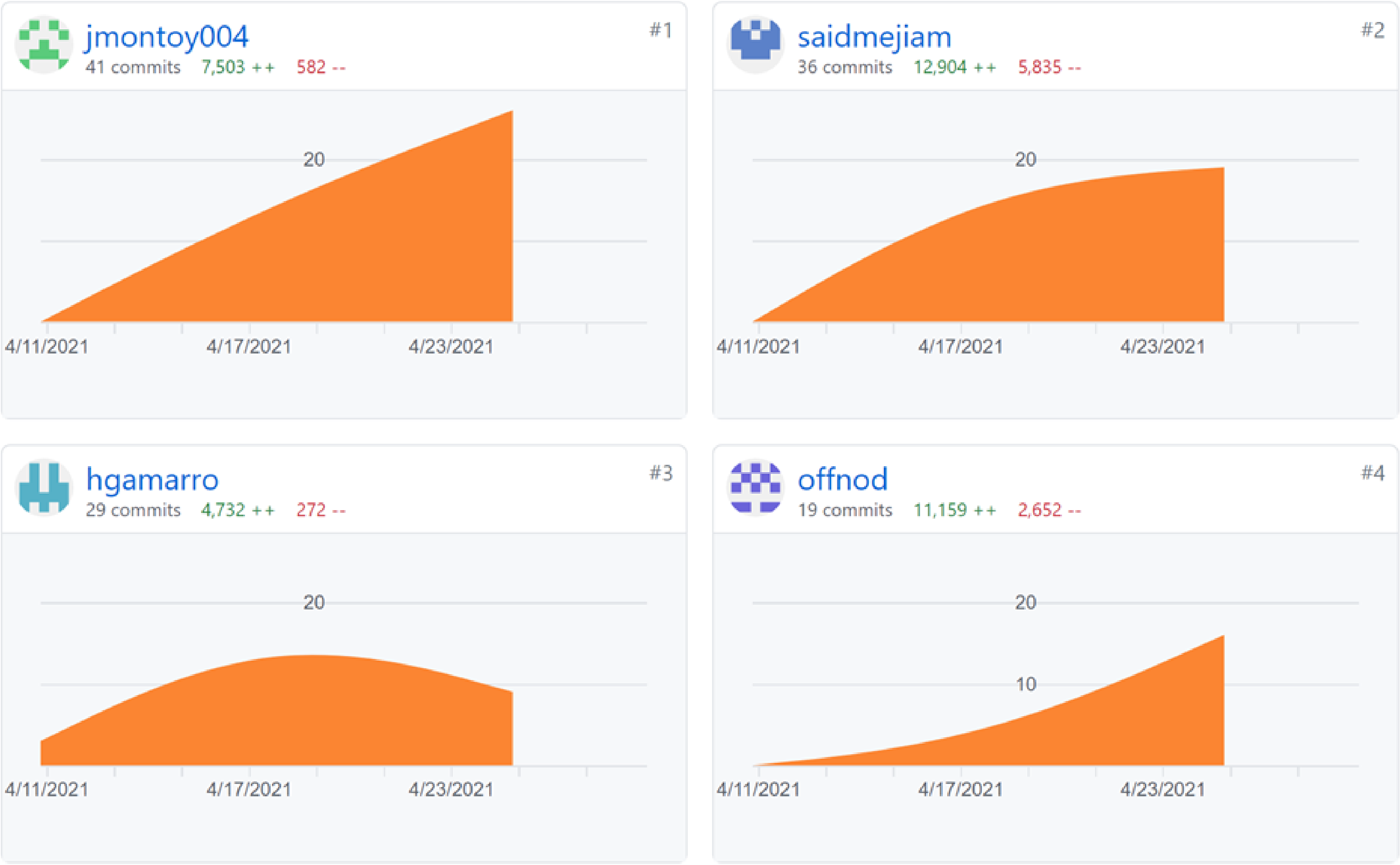
- Image (64x64x3)
- Using SegNet



```
def segnet(input_shape, n_labels, kernel=3, pool_size=(2, 2), output_mode="softmax"):
    # encoder
    inputs = Input(shape=input_shape)

    conv_1 = Convolution2D(64, (kernel, kernel), padding="same")(inputs)
    conv_1 = BatchNormalization()(conv_1)
    conv_1 = Activation("relu")(conv_1)
    conv_2 = Convolution2D(64, (kernel, kernel), padding="same")(conv_1)
    conv_2 = BatchNormalization()(conv_2)
    conv_2 = Activation("relu")(conv_2)

    pool_1, mask_1 = MaxPoolingWithArgmax2D(pool_size)(conv_2)

    conv_3 = Convolution2D(128, (kernel, kernel), padding="same")(pool_1)
    conv_3 = BatchNormalization()(conv_3)
    conv_3 = Activation("relu")(conv_3)
    conv_4 = Convolution2D(128, (kernel, kernel), padding="same")(conv_3)
    conv_4 = BatchNormalization()(conv_4)
    conv_4 = Activation("relu")(conv_4)

    pool_2, mask_2 = MaxPoolingWithArgmax2D(pool_size)(conv_4)

    conv_5 = Convolution2D(256, (kernel, kernel), padding="same")(pool_2)
    conv_5 = BatchNormalization()(conv_5)
    conv_5 = Activation("relu")(conv_5)
    conv_6 = Convolution2D(256, (kernel, kernel), padding="same")(conv_5)
    conv_6 = BatchNormalization()(conv_6)
    conv_6 = Activation("relu")(conv_6)
    conv_7 = Convolution2D(256, (kernel, kernel), padding="same")(conv_6)
    conv_7 = BatchNormalization()(conv_7)
    conv_7 = Activation("relu")(conv_7)

    pool_3, mask_3 = MaxPoolingWithArgmax2D(pool_size)(conv_7)

    conv_8 = Convolution2D(512, (kernel, kernel), padding="same")(pool_3)
    conv_8 = BatchNormalization()(conv_8)
    conv_8 = Activation("relu")(conv_8)
    conv_9 = Convolution2D(512, (kernel, kernel), padding="same")(conv_8)
    conv_9 = BatchNormalization()(conv_9)
    conv_9 = Activation("relu")(conv_9)
    conv_10 = Convolution2D(512, (kernel, kernel), padding="same")(conv_9)
    conv_10 = BatchNormalization()(conv_10)
    conv_10 = Activation("relu")(conv_10)

    pool_4, mask_4 = MaxPoolingWithArgmax2D(pool_size)(conv_10)

    conv_11 = Convolution2D(512, (kernel, kernel), padding="same")(pool_4)
    conv_11 = BatchNormalization()(conv_11)
    conv_11 = Activation("relu")(conv_11)
    conv_12 = Convolution2D(512, (kernel, kernel), padding="same")(conv_11)
    conv_12 = BatchNormalization()(conv_12)
    conv_12 = Activation("relu")(conv_12)
    conv_13 = Convolution2D(512, (kernel, kernel), padding="same")(conv_12)
    conv_13 = BatchNormalization()(conv_13)
    conv_13 = Activation("relu")(conv_13)
```

```
# decoder

unpool_1 = MaxUnpooling2D(pool_size)([pool_5, mask_5])

conv_14 = Convolution2D(512, (kernel, kernel), padding="same")(unpool_1)
conv_14 = BatchNormalization()(conv_14)
conv_14 = Activation("relu")(conv_14)
conv_15 = Convolution2D(512, (kernel, kernel), padding="same")(conv_14)
conv_15 = BatchNormalization()(conv_15)
conv_15 = Activation("relu")(conv_15)
conv_16 = Convolution2D(512, (kernel, kernel), padding="same")(conv_15)
conv_16 = BatchNormalization()(conv_16)
conv_16 = Activation("relu")(conv_16)

unpool_2 = MaxUnpooling2D(pool_size)([conv_16, mask_4])

conv_17 = Convolution2D(512, (kernel, kernel), padding="same")(unpool_2)
conv_17 = BatchNormalization()(conv_17)
conv_17 = Activation("relu")(conv_17)
conv_18 = Convolution2D(512, (kernel, kernel), padding="same")(conv_17)
conv_18 = BatchNormalization()(conv_18)
conv_18 = Activation("relu")(conv_18)
conv_19 = Convolution2D(256, (kernel, kernel), padding="same")(conv_18)
conv_19 = BatchNormalization()(conv_19)
conv_19 = Activation("relu")(conv_19)

unpool_3 = MaxUnpooling2D(pool_size)([conv_19, mask_3])

conv_20 = Convolution2D(256, (kernel, kernel), padding="same")(unpool_3)
conv_20 = BatchNormalization()(conv_20)
conv_20 = Activation("relu")(conv_20)
conv_21 = Convolution2D(256, (kernel, kernel), padding="same")(conv_20)
conv_21 = BatchNormalization()(conv_21)
conv_21 = Activation("relu")(conv_21)
conv_22 = Convolution2D(128, (kernel, kernel), padding="same")(conv_21)
conv_22 = BatchNormalization()(conv_22)
conv_22 = Activation("relu")(conv_22)

unpool_4 = MaxUnpooling2D(pool_size)([conv_22, mask_2])

conv_23 = Convolution2D(128, (kernel, kernel), padding="same")(unpool_4)
conv_23 = BatchNormalization()(conv_23)
conv_23 = Activation("relu")(conv_23)
conv_24 = Convolution2D(64, (kernel, kernel), padding="same")(conv_23)
conv_24 = BatchNormalization()(conv_24)
conv_24 = Activation("relu")(conv_24)

unpool_5 = MaxUnpooling2D(pool_size)([conv_24, mask_1])

conv_25 = Convolution2D(64, (kernel, kernel), padding="same")(unpool_5)
conv_25 = BatchNormalization()(conv_25)
conv_25 = Activation("relu")(conv_25)

conv_26 = Convolution2D(n_labels, (1, 1), padding="valid")(conv_25)
conv_26 = BatchNormalization()(conv_26)
conv_26 = Reshape(
    (input_shape[0] * input_shape[1], n_labels),
    input_shape=(input_shape[0], input_shape[1], n_labels),
)(conv_26)

outputs = Activation(output_mode)(conv_26)
#print("Build decoder done..")

model = Model(inputs=inputs, outputs=outputs, name="SegNet")

return model
```

Git hub commits 151 total



jmontoy004 #1
41 commits 7,503 ++ 582 --

saidmejiam #2
36 commits 12,904 ++ 5,835 --

hgamarro #3
29 commits 4,732 ++ 272 --

offnod #4
19 commits 11,159 ++ 2,652 --

Notebook total

- HG: 4
- JB: 7
- JM: 4
- SM: 3