Retrieve Roads from Aerial Imagery Using Deep Learning

Capstone Project Mid-Point Presentation, CPLN 680, Spring 2022, UPenn

Jiamin Tan

Object

Learn how to train a neural network to detect roads from aerial images.

Object

<u>Learn</u> how to train a neural network to detect roads from aerial images.

Current Status:
I am learning ☺
but my algorithm is not learning ☹

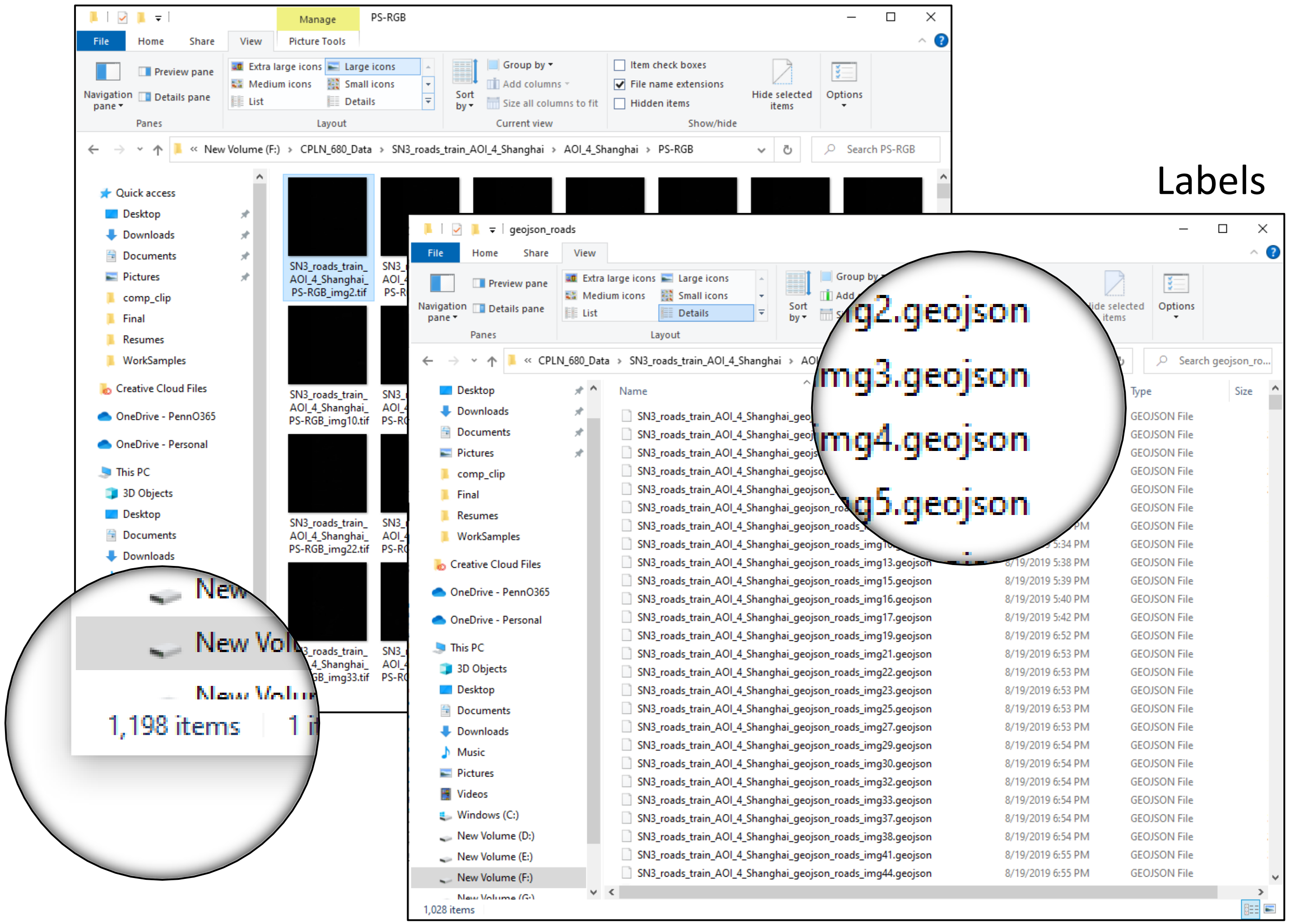
which means both of us haven't learnt enough so far…

Main Data Source

https://spacenet.ai/

Training Image Set

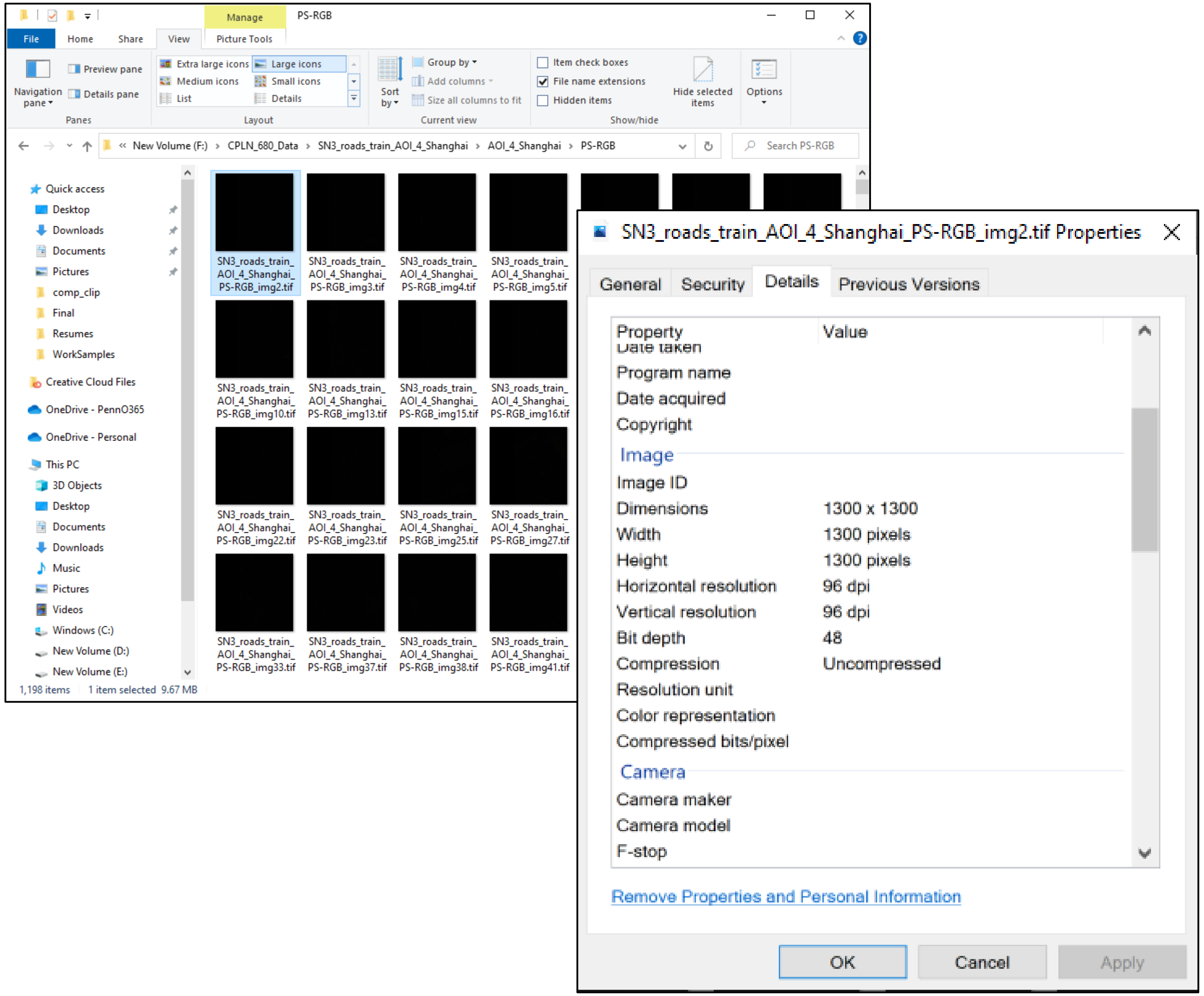
Labels



.tif files stored satellite images.

.geojson files stored strings (roads) as ground truth.

Training Image Set



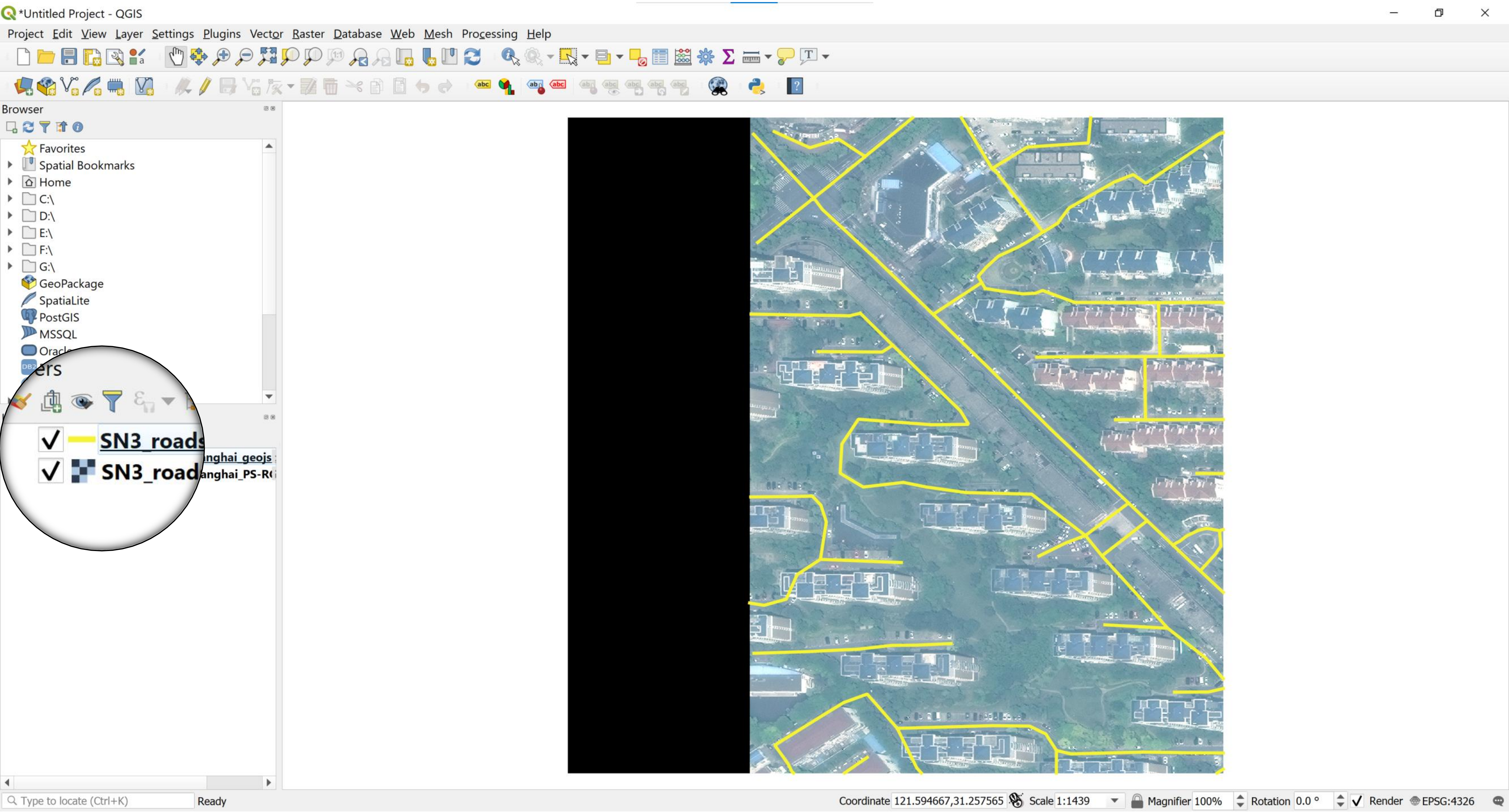
For each pan-sharpened Image:

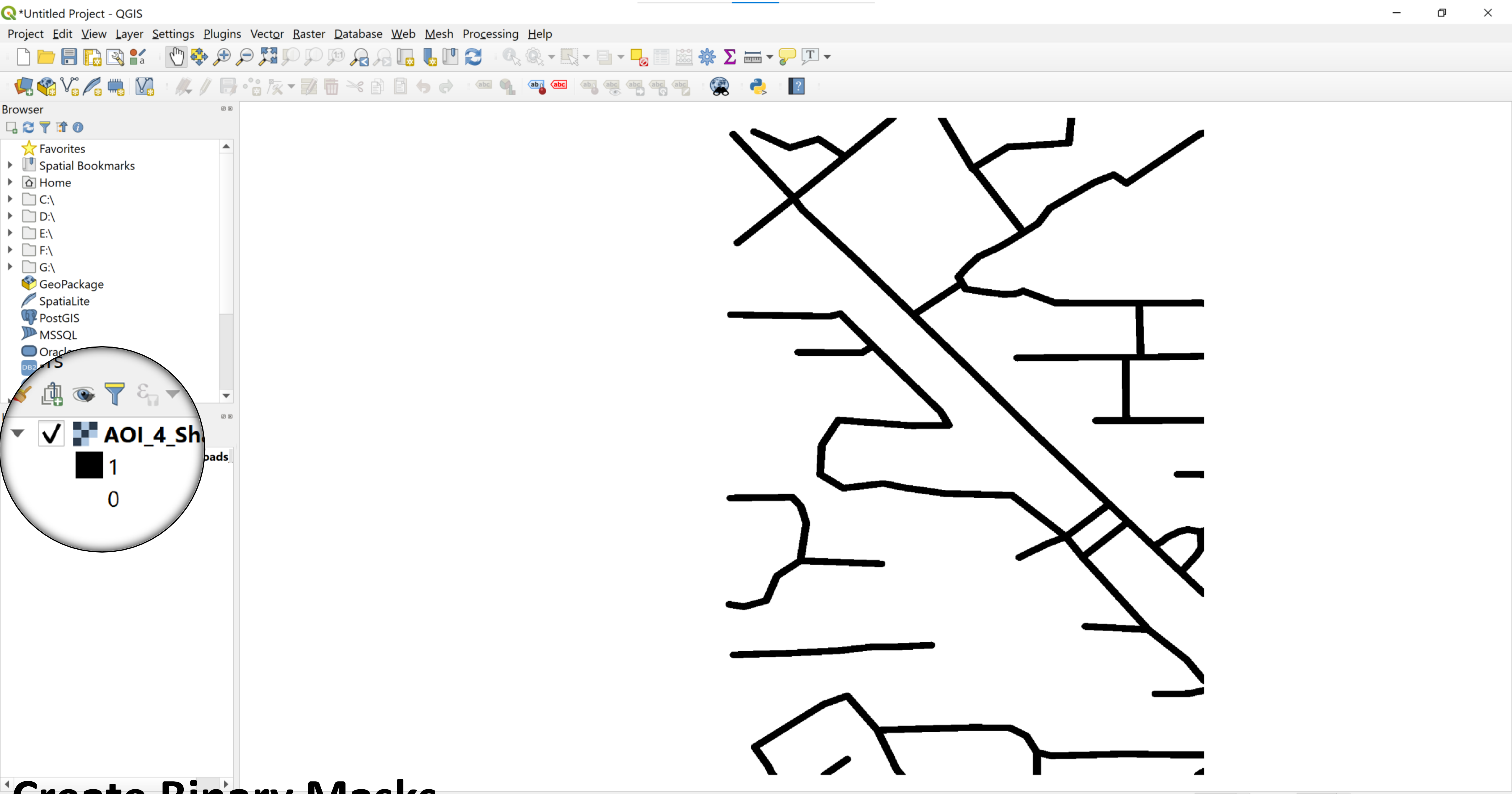
1300 x 1300 pixels.

Each pixel has a spatial resolution of 0.31m x 0.31m.

Each tile is, therefore, 400m x 400m.

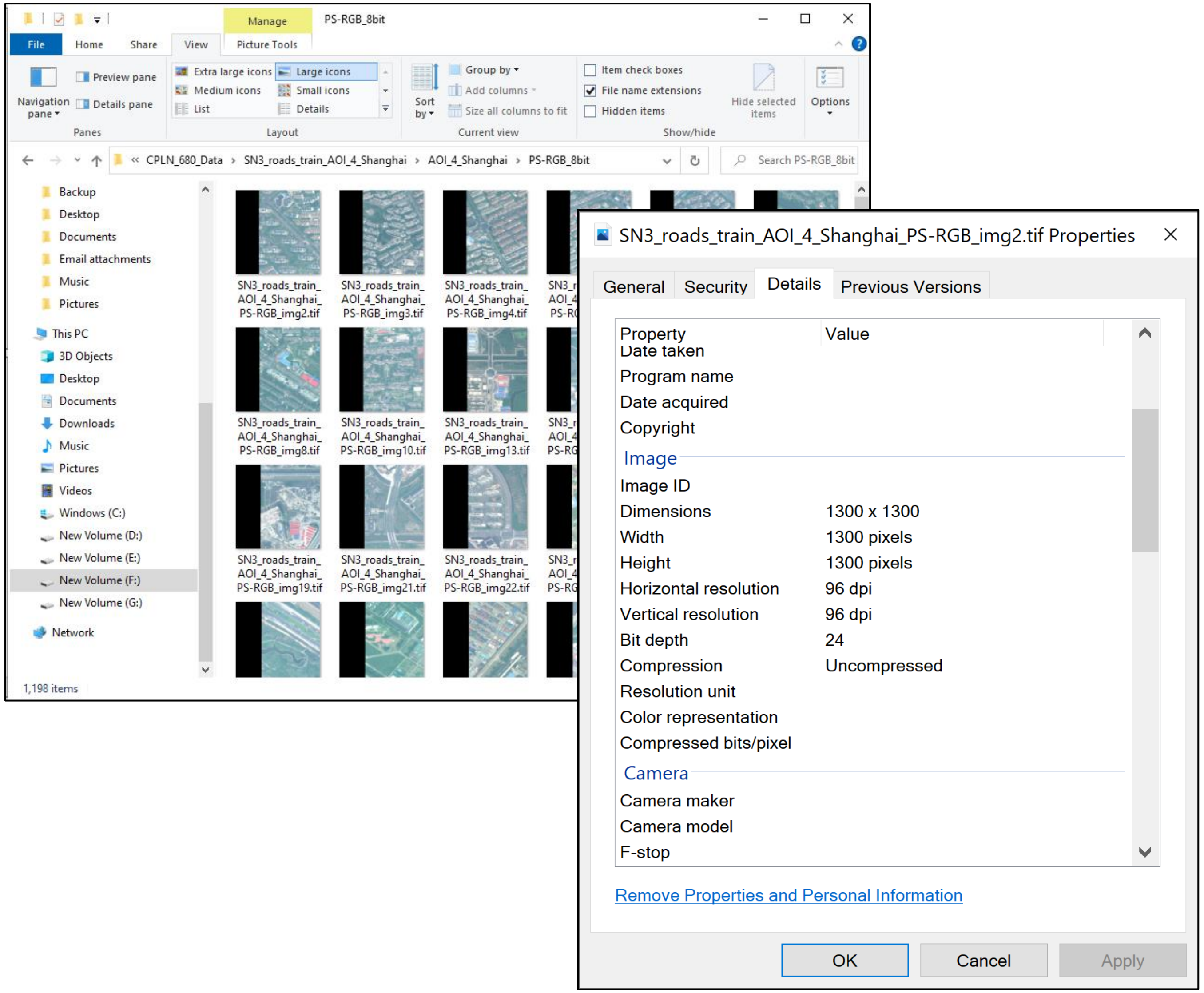
Bit depth is 48, so 16 bit for each band, and the value of a pixel in each band is from 0 to 65535.


*Untitled Project - QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Web Mesh Processing Help
Browser
Favorites
Spatial Bookmarks
Home
C:\
D:\
E:\
F:\
G:\
GeoPackage
SpatiaLite
PostGIS
MSSQL
SN3_roads
SN3_road
Type to locate (Ctrl+K)
Ready
Coordinate 121.594667,31.257565
Scale 1:1439
Magnifier 100%
Rotation 0.0 °
Render
EPSG:4326



Create Binary Masks

Training Image Set (8-bit)



Each Image **Now**:

1300 x 1300 pixels.

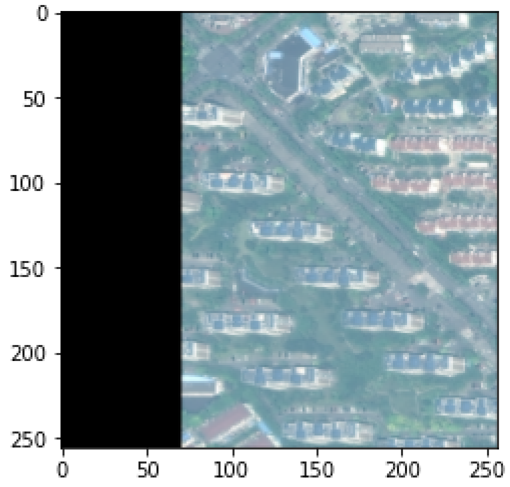
Each pixel has a spatial resolution of 0.31m x 0.31m.

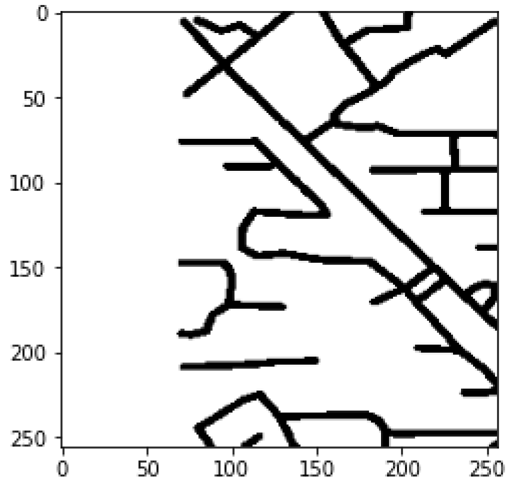
Each tile is, therefore, 400m x 400m.

Bit depth is **24**, so **8** bit for each band – value of a pixel in each band is from **0** to **255**.

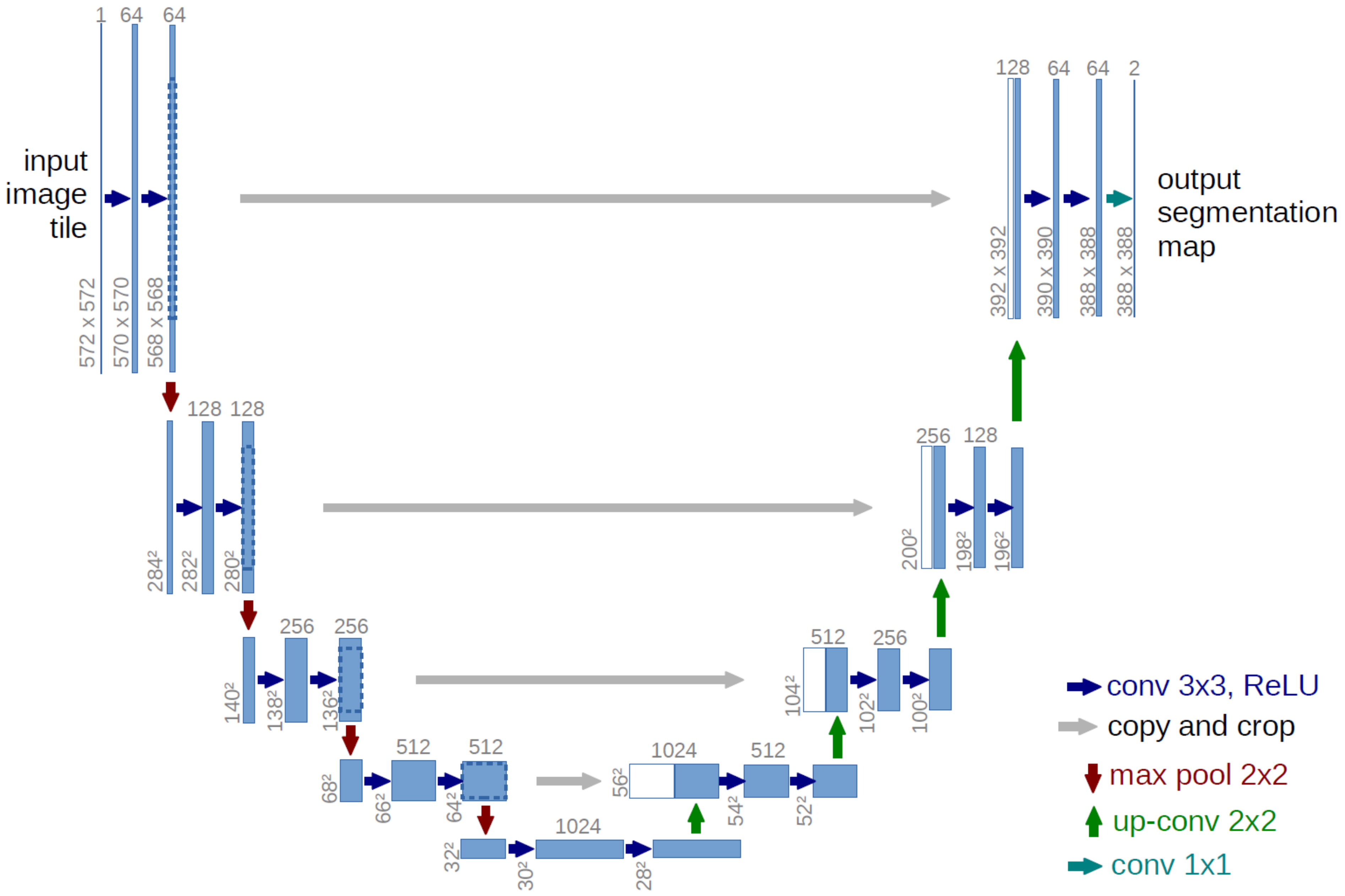
Create 8-Bit Images

```python
image_x = random.randint(0, len(train_img_ids))
plt.imshow(X_train[train_img_ids.index('SN3_roads_train_AOI_4_Shanghai_PS-RGB_img2.tif')])
plt.show()
plt.imshow(np.squeeze(Y_train[train_img_ids.index('SN3_roads_train_AOI_4_Shanghai_PS-RGB_img2.tif')]), cmap='Greys')
plt.show()
```





Load Training Set into Python



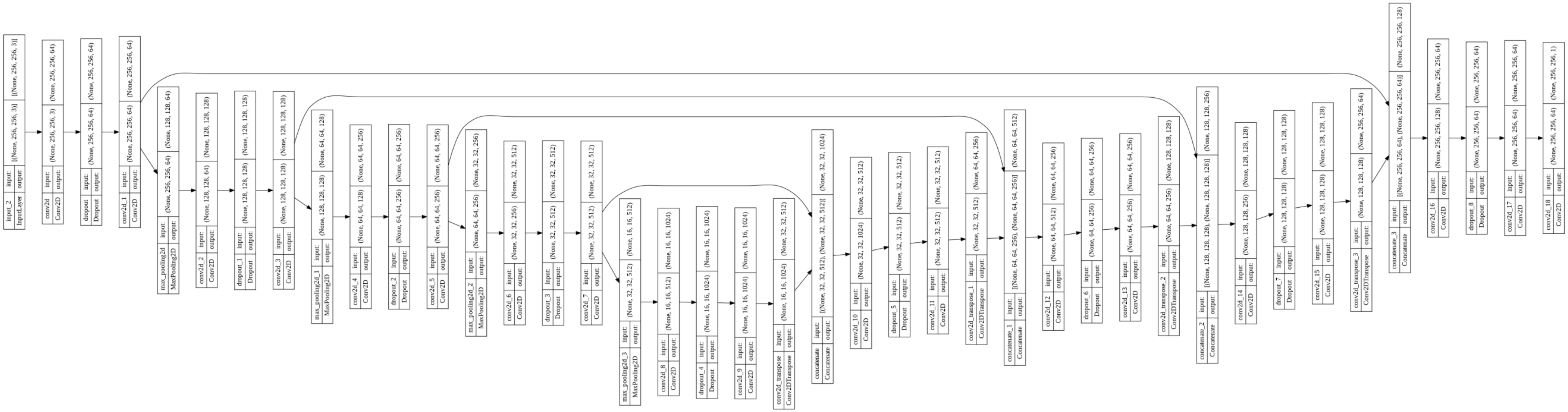
U-Net by Ronneberger et al. (2015)

https://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a/

```
[19] c1 = tf.keras.layers.Conv2D(64, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(inputs)
     c1 = tf.keras.layers.Dropout(0.1)(c1)
     c1 = tf.keras.layers.Conv2D(64, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(c1)
     p1 = tf.keras.layers.MaxPooling2D((2,2))(c1)

[20] c2 = tf.keras.layers.Conv2D(128, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(p1)
     c2 = tf.keras.layers.Dropout(0.1)(c2)
     c2 = tf.keras.layers.Conv2D(128, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(c2)
     p2 = tf.keras.layers.MaxPooling2D((2,2))(c2)

[21] c3 = tf.keras.layers.Conv2D(256, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(p2)
     c3 = tf.keras.layers.Dropout(0.1)(c3)
     c3 = tf.keras.layers.Conv2D(256, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(c3)
     p3 = tf.keras.layers.MaxPooling2D((2,2))(c3)

[22] c4 = tf.keras.layers.Conv2D(512, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(p3)
     c4 = tf.keras.layers.Dropout(0.1)(c4)
     c4 = tf.keras.layers.Conv2D(512, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(c4)
     p4 = tf.keras.layers.MaxPooling2D((2,2))(c4)

[23] c5 = tf.keras.layers.Conv2D(1024, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(p4)
     c5 = tf.keras.layers.Dropout(0.3)(c5)
     c5 = tf.keras.layers.Conv2D(1024, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(c5)

[24] u6 = tf.keras.layers.Convolution2DTranspose(512, (2,2), strides = (2,2), padding = 'same')(c5)
     u6 = tf.keras.layers.concatenate([u6, c4])
     c6 = tf.keras.layers.Conv2D(512, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(u6)
     c6 = tf.keras.layers.Dropout(0.2)(c6)
     c6 = tf.keras.layers.Conv2D(512, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(c6)

[25] u7 = tf.keras.layers.Convolution2DTranspose(256, (2,2), strides = (2,2), padding = 'same')(c6)
     u7 = tf.keras.layers.concatenate([u7, c3])
     c7 = tf.keras.layers.Conv2D(256, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(u7)
     c7 = tf.keras.layers.Dropout(0.2)(c7)
     c7 = tf.keras.layers.Conv2D(256, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(c7)

[26] u8 = tf.keras.layers.Convolution2DTranspose(128, (2,2), strides = (2,2), padding = 'same')(c7)
     u8 = tf.keras.layers.concatenate([u8, c2])
     c8 = tf.keras.layers.Conv2D(128, (16,1), activation = 'relu', kernel_initializer = 'he_normal', padding = 'same')(u8)
```

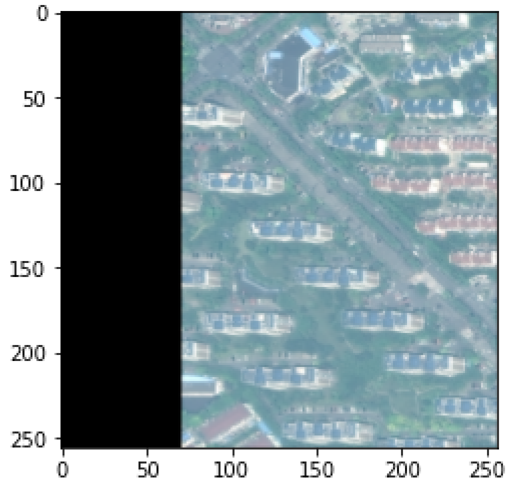
Hard Code U-Net Using Keras

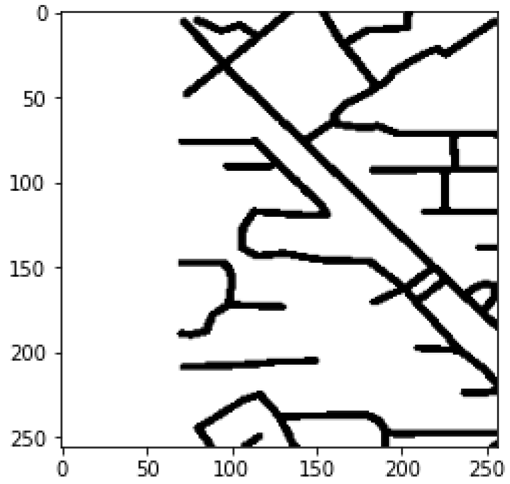
Codes learnt from: Sreenivas Bhattiprolu, https://www.youtube.com/playlist?list=PLZsOBAyNTZwbR08R959iCvYT3qzhxvGOE



Hard Code U-Net Using Keras (Cont.)

```python
image_x = random.randint(0, len(train_img_ids))
plt.imshow(X_train[train_img_ids.index('SN3_roads_train_AOI_4_Shanghai_PS-RGB_img2.tif')])
plt.show()
plt.imshow(np.squeeze(Y_train[train_img_ids.index('SN3_roads_train_AOI_4_Shanghai_PS-RGB_img2.tif')]), cmap='Greys')
plt.show()
```





Each Image **After Resizing**:

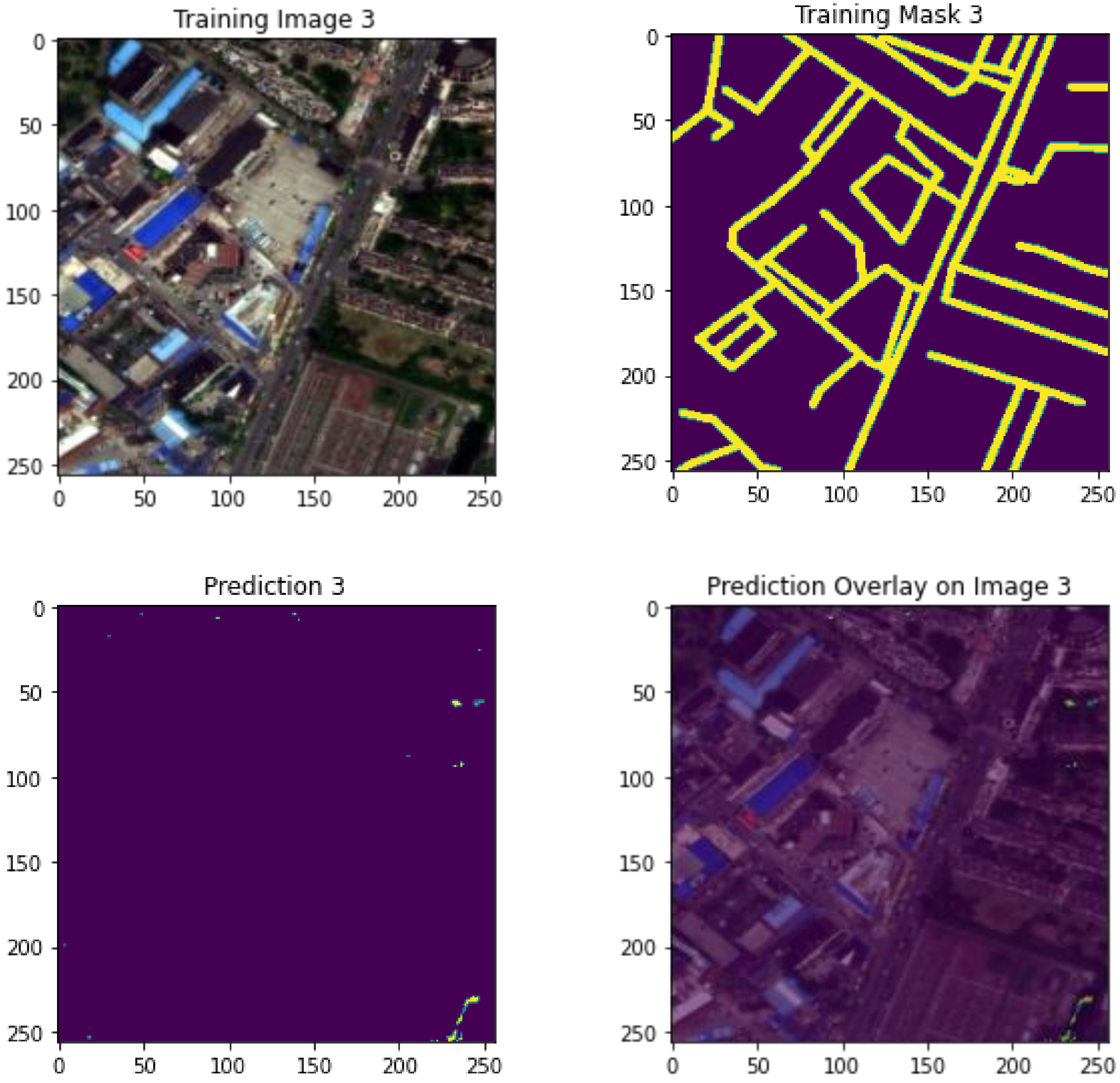
256 x **256** pixels.

Each pixel has a spatial resolution of **1.56**m x **1.56**m.

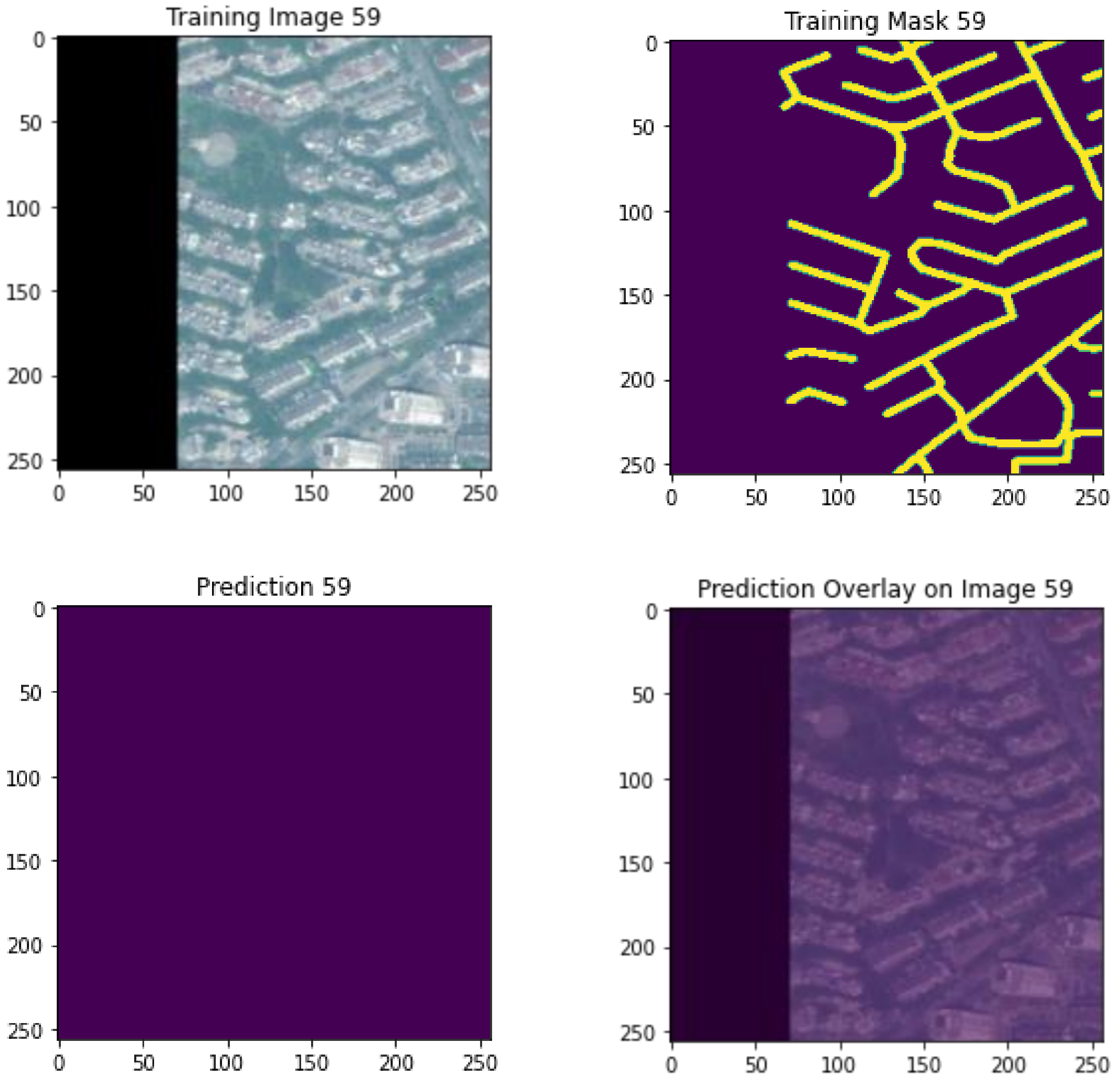
Each tile is, still, 400m x 400m.

Bit depth is **24**, so **8** bit for each band – value of a pixel in each band is from **0** to **255**.

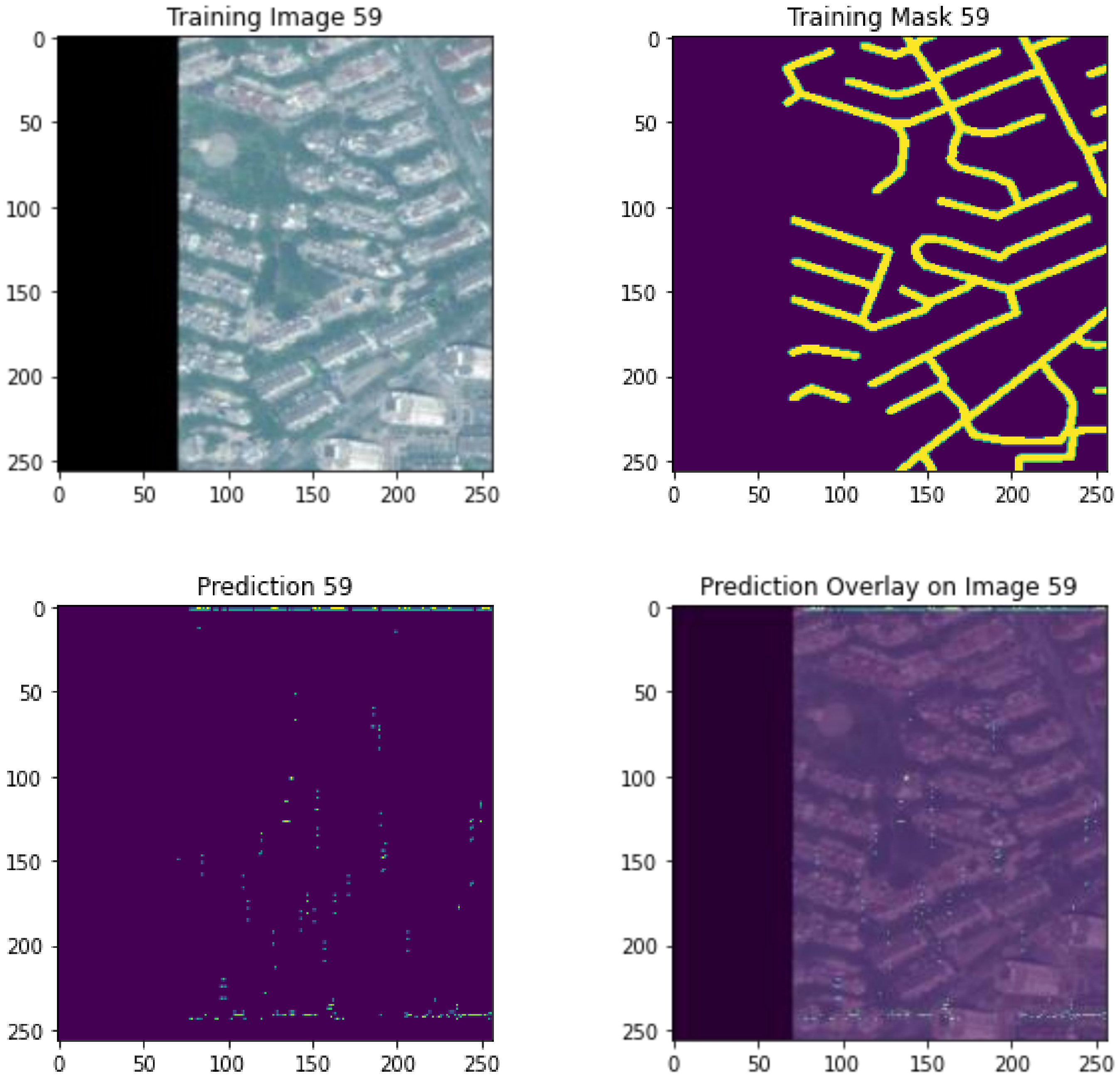
Train 100 images with reduced size (256, 256)



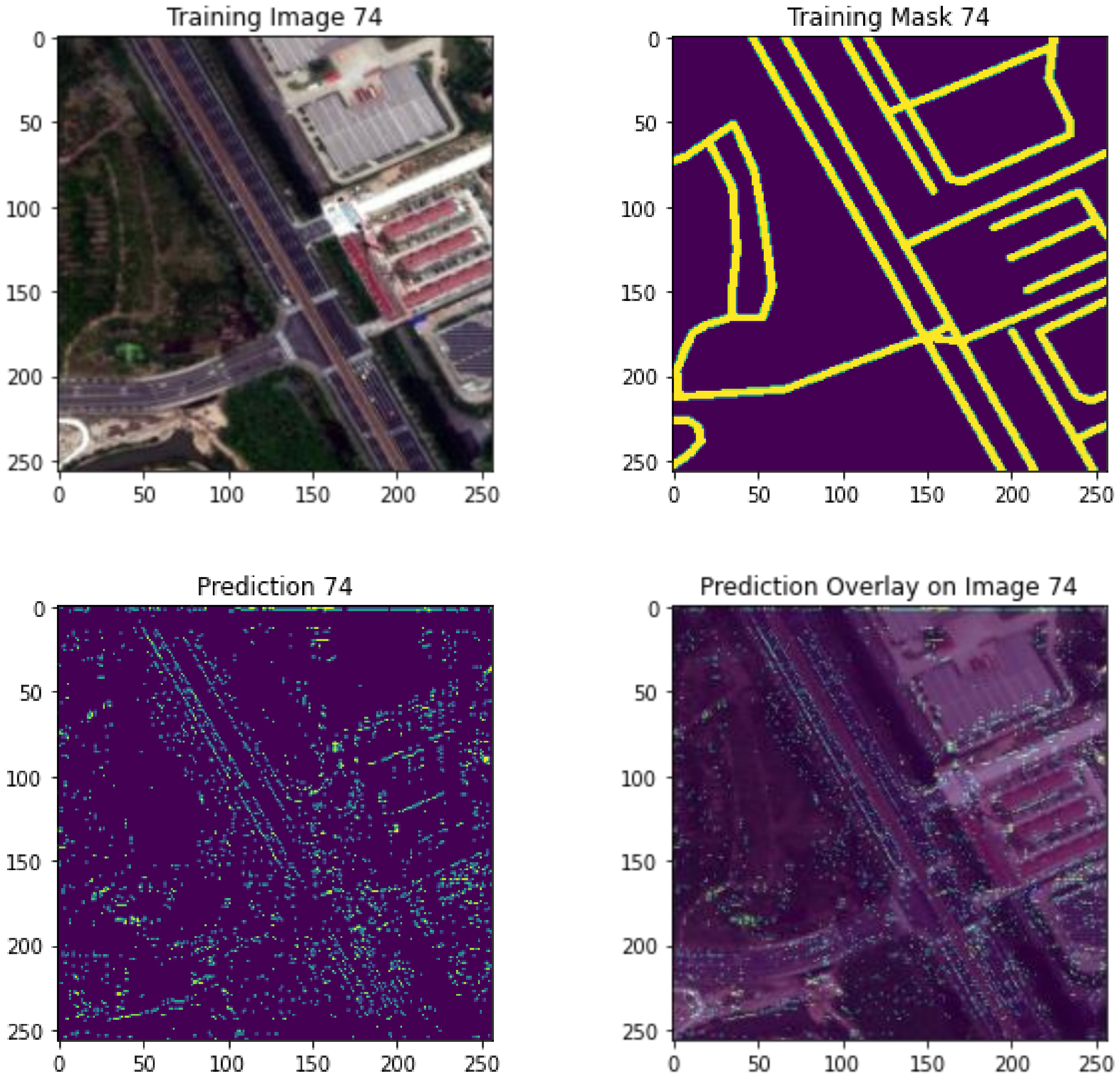
The results are not good (threshold = 0.5)…



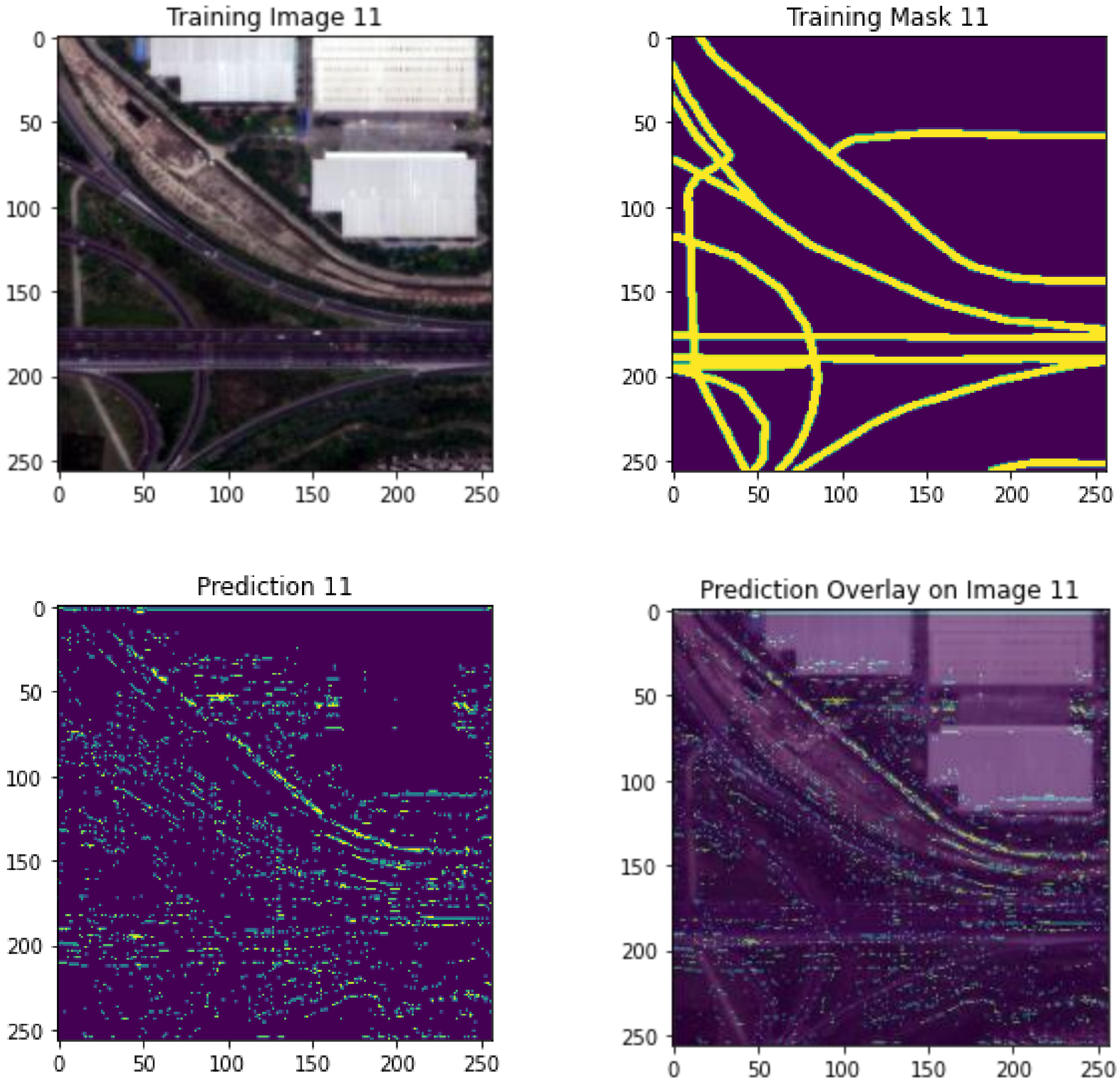
The results are not good (threshold = 0.5)…



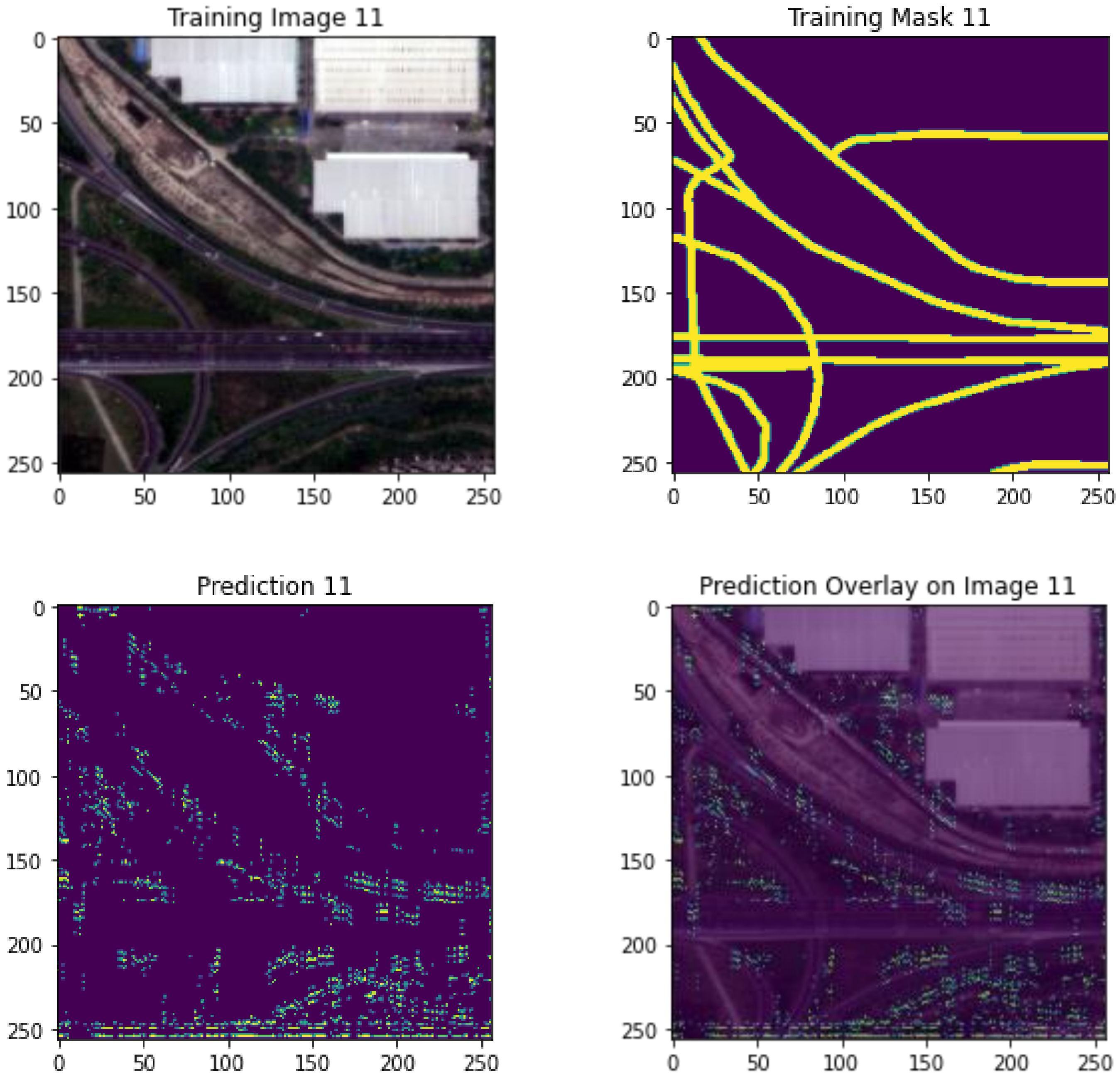
The same training image in another run (threshold = 0.5)…



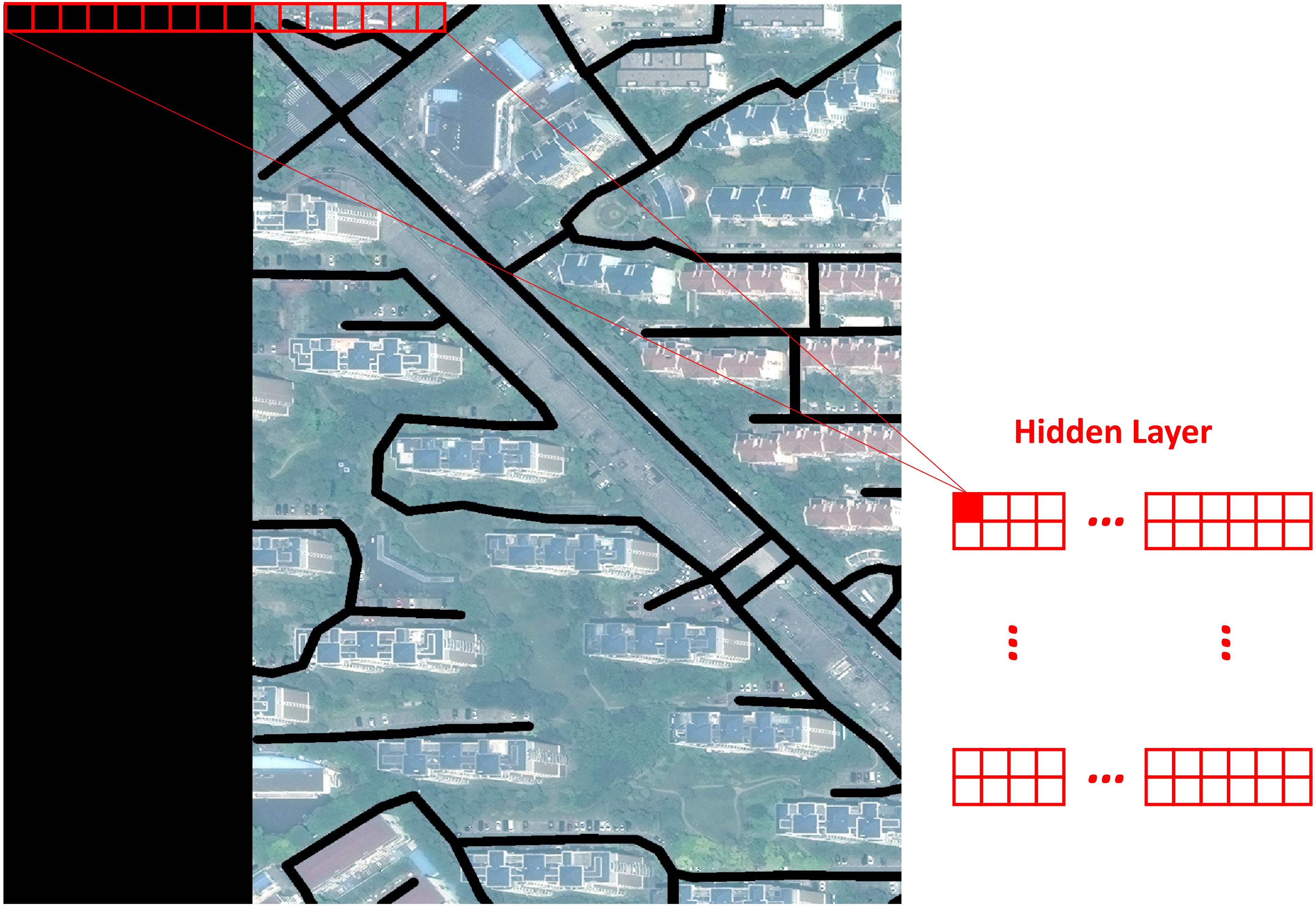
Sometimes it sort of learnt (threshold = 0.5)…



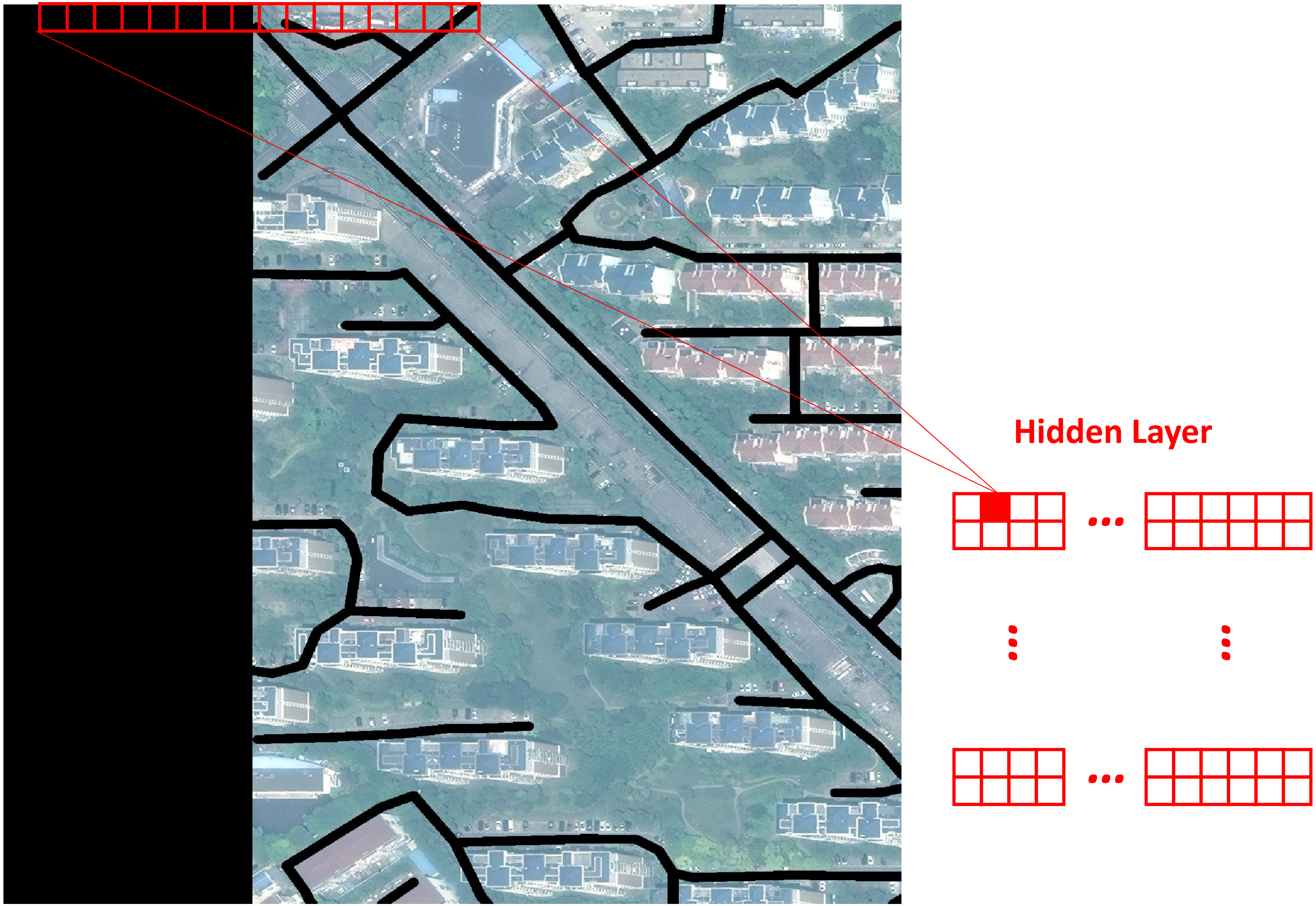
Sometimes it sort of learnt (threshold = 0.5)…



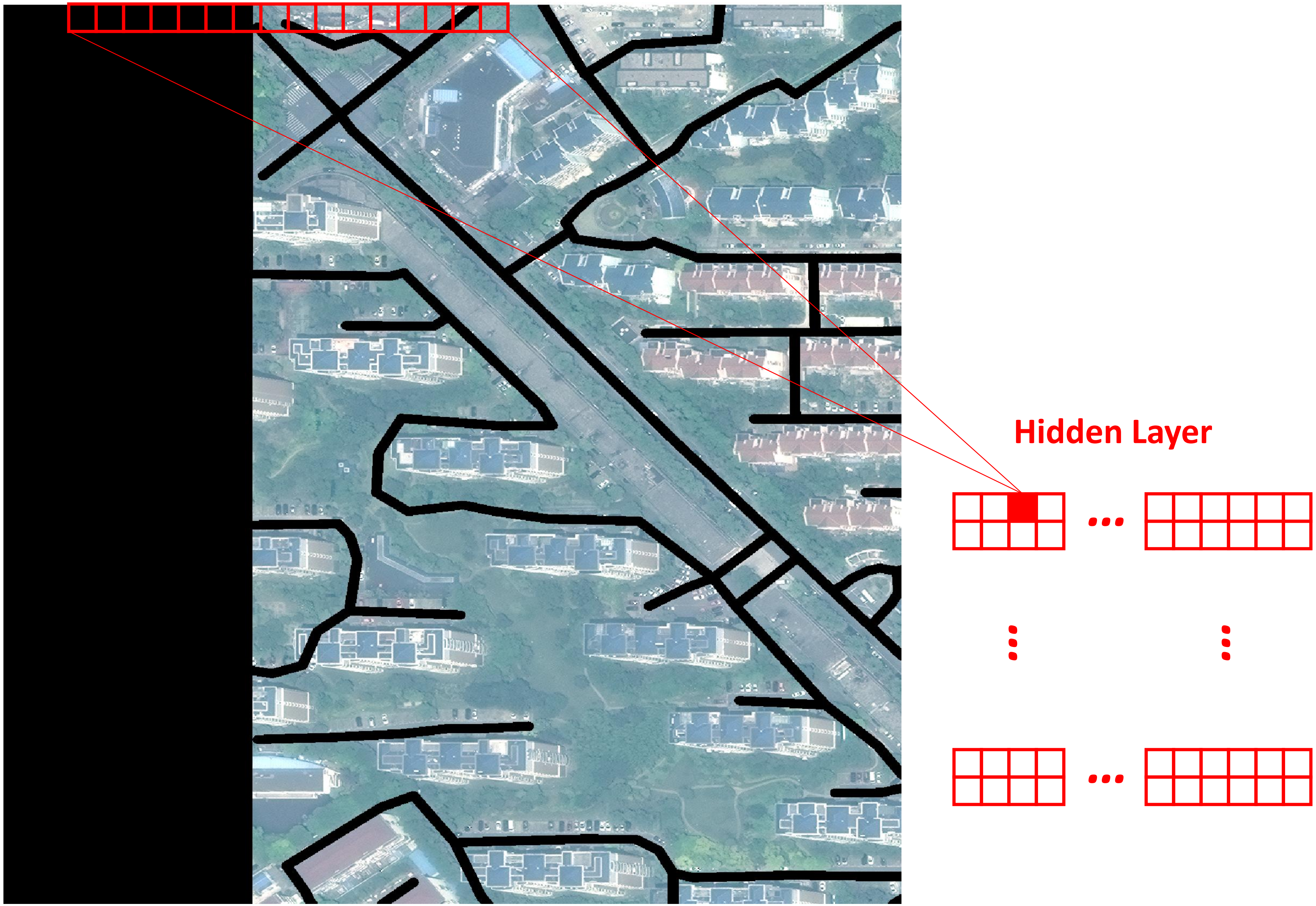
Or it learnt the wrong thing in another run (threshold = 0.3)…



A (16, 1) instead of (3, 3) Kernel *(not in real scale)*



A (16, 1) instead of (3, 3) Kernel *(not in real scale)*



A (16, 1) instead of (3, 3) Kernel *(not in real scale)*

Some Directions for the Next Step

Rotate the images for augmentation

Figure out a suitable kernel size

Create buffers by road type

Learn about transfer learning which uses pre-trained models

Read more papers on road segmentation

Try to train more images