In [1]:
```
import os
import cv2
import json
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from PIL import Image
from skimage.transform import resize
from sklearn.model_selection import train_test_split
%matplotlib inline
```

In [3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call driv
e.mount("/content/drive", force_remount=True).
```

In [4]:
```
images_dir ='/content/drive/MyDrive/png/train'
masks_dir = '/content/drive/MyDrive/png/train_labels'
val_images_dir ='/content/drive/MyDrive/png/val'
val_masks_dir = '/content/drive/MyDrive/png/val_labels'
test_images_dir='/content/drive/MyDrive/png/test'
test_masks_dir='/content/drive/MyDrive/png/test_labels'
```

In [5]:
```
images_listdir = os.listdir(images_dir)
images_listdir.sort()
masks_listdir = os.listdir(masks_dir)
masks_listdir.sort()
random_images = np.random.choice(masks_listdir, size = 9, replace = False)
random_images.sort()
test_images_listdir = os.listdir(test_images_dir)
test_images_listdir.sort()
test_masks_listdir = os.listdir(test_masks_dir)
test_masks_listdir.sort()
```

In [6]:
```
print(len(images_listdir))
print(len(masks_listdir))
print(len(test_images_listdir))
print(len(test_masks_listdir))
```

```
137
137
10
10
```

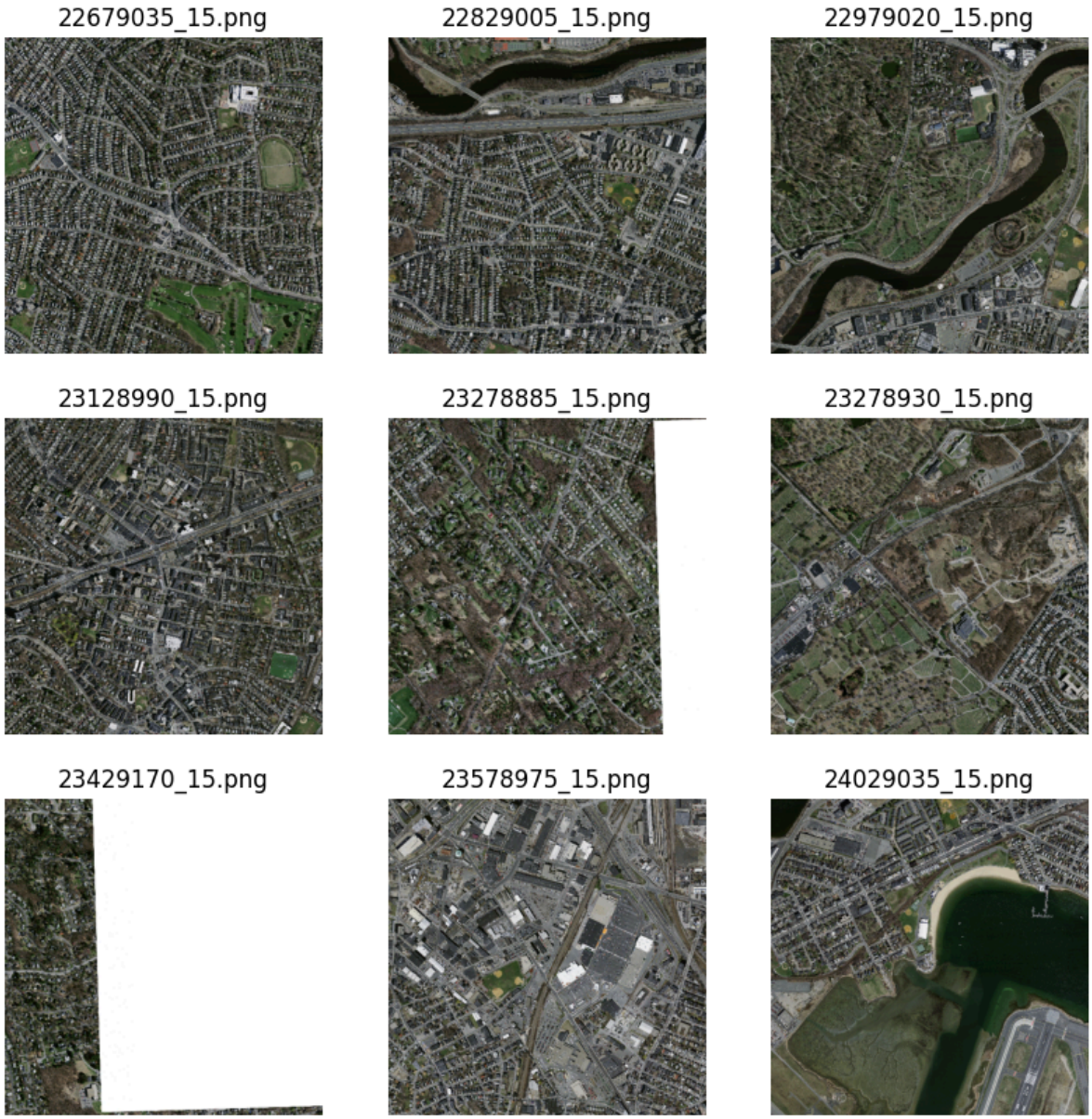
In [7]:
```
image_size=512
input_image_size=(512,512)
```

In [8]:
```
def read_image(path):
    img = cv2.imread(path)
    img = cv2.resize(img, (image_size, image_size))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return img
```

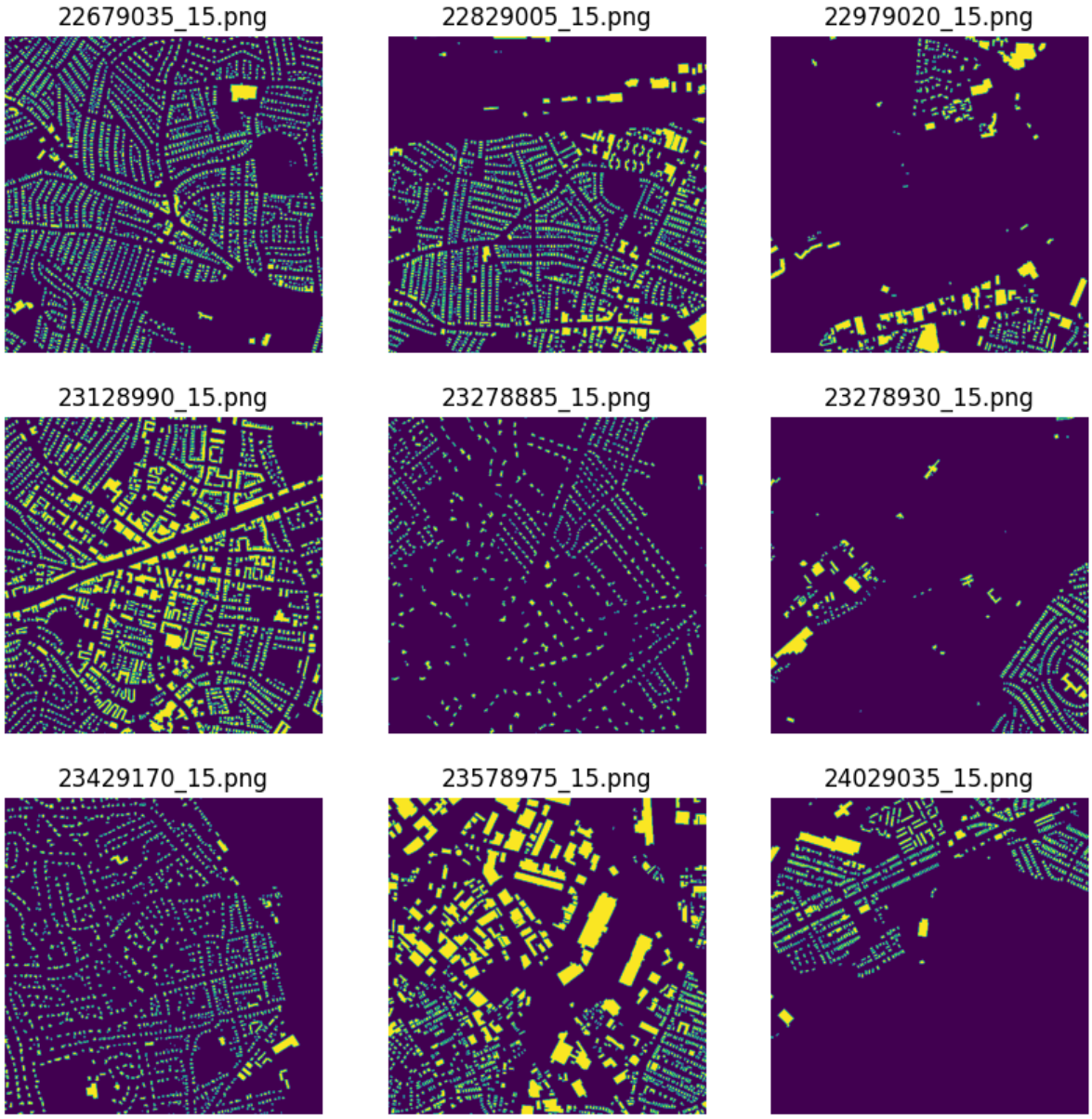
In [37]:
```
number = 30
```

In [38]:
```
rows = 3
cols = 3
```

```
fig, ax = plt.subplots(rows, cols, figsize = (10,10))
for i, ax in enumerate(ax.flat):
    if i < len(random_images):
        img = read_image(f"{images_dir}/{random_images[i]}")
        ax.set_title(f"{random_images[i]}")
        ax.imshow(img)
        ax.axis('off')
```



In [39]:
```
fig, ax = plt.subplots(rows, cols, figsize = (10,10))
for i, ax in enumerate(ax.flat):
    if i < len(random_images):
        file=random_images[i]
        if os.path.exists(os.path.join(masks_dir,file)):
            img = read_image(f"{masks_dir}/{file}")
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            ax.set_title(f"{random_images[i]}")
            ax.imshow(img)
            ax.axis('off')
        else:
            print('not exist')
```



In [40]:
```
MASKS=np.zeros((1,image_size, image_size, 1), dtype=bool)
IMAGES=np.zeros((1,image_size, image_size, 3),dtype=np.uint8)

for j,file in enumerate(masks_listdir[0:number]):   ##the smaller, the faster
    try:
        image = read_image(f"{images_dir}/{file}")
        image_ex = np.expand_dims(image, axis=0)
        IMAGES = np.vstack([IMAGES, image_ex])
        file2=file[0:-4]+'.png'
        mask = read_image(f"{masks_dir}/{file2}")
        mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
        mask = mask.reshape(512,512,1)
        mask_ex = np.expand_dims(mask, axis=0)
        MASKS = np.vstack([MASKS, mask_ex])
    except:
        print(file)
        continue
```

In [41]:
```
TMASKS=np.zeros((1,image_size, image_size,1), dtype=bool)
TIMAGES=np.zeros((1,image_size, image_size,3),dtype=np.uint8)

for j,file in enumerate(test_images_listdir):
    try:
        image = read_image(f"{test_images_dir}/{file}")
        image_ex = np.expand_dims(image, axis=0)
```

```
        TIMAGES = np.vstack([TIMAGES, image_ex])
        file2=file[0:-4]+'.png'
        mask = read_image(f"{test_masks_dir}/{file2}")
        mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
        mask = mask.reshape(512,512,1)
        mask_ex = np.expand_dims(mask, axis=0)
        TMASKS = np.vstack([TMASKS, mask_ex])
    except:
        print(file)
        continue
```

In [42]:
```
images=np.array(IMAGES)[1:number+1]
masks=np.array(MASKS)[1:number+1]
print(images.shape,masks.shape)

test_images=np.array(TIMAGES)[1:]
test_masks=np.array(TMASKS)[1:]
print(test_images.shape,test_masks.shape)
```

```
(30, 512, 512, 3) (30, 512, 512, 1)
(10, 512, 512, 3) (10, 512, 512, 1)
```

In [43]:
```
images_train, images_test, masks_train, masks_test = train_test_split(images, masks
print(len(images_train), len(masks_train))
```

```
24 24
```

In [44]:
```
def conv_block(input, num_filters):
    conv = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(input)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation("relu")(conv)
    conv = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(conv)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation("relu")(conv)
    return conv

def encoder_block(input, num_filters):
    skip = conv_block(input, num_filters)
    pool = tf.keras.layers.MaxPool2D((2,2))(skip)
    return skip, pool

def decoder_block(input, skip, num_filters):
    up_conv = tf.keras.layers.Conv2DTranspose(num_filters, (2,2), strides=2, paddir
    conv = tf.keras.layers.Concatenate()([up_conv, skip])
    conv = conv_block(conv, num_filters)
    return conv

def Unet(input_shape):
    inputs = tf.keras.layers.Input(input_shape)

    skip1, pool1 = encoder_block(inputs, 64)
    skip2, pool2 = encoder_block(pool1, 128)
    skip3, pool3 = encoder_block(pool2, 256)
    skip4, pool4 = encoder_block(pool3, 512)

    bridge = conv_block(pool4, 1024)

    decode1 = decoder_block(bridge, skip4, 512)
    decode2 = decoder_block(decode1, skip3, 256)
    decode3 = decoder_block(decode2, skip2, 128)
    decode4 = decoder_block(decode3, skip1, 64)
    outputs = tf.keras.layers.Conv2D(1, 1, padding="same", activation="sigmoid")(de
    model = tf.keras.models.Model(inputs, outputs, name="U-Net")
    return model
```

```
unet_model = Unet((512,512,3))
unet_model.compile(optimizer=Adam(learning_rate=0.0001), loss= 'binary_crossentropy
unet_model.summary()
```

```
Model: "U-Net"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_4 (InputLayer)        [(None, 512, 512, 3)]     0         []

 conv2d_57 (Conv2D)          (None, 512, 512, 64)      1792      ['input_4[0]
[0]']

 batch_normalization_54 (Ba  (None, 512, 512, 64)      256       ['conv2d_57[0]
[0]']
 tchNormalization)

 activation_54 (Activation)  (None, 512, 512, 64)      0         ['batch_normal
ization_54[0][0]
                                                                  ']

 conv2d_58 (Conv2D)          (None, 512, 512, 64)      36928     ['activation_5
4[0][0]']

 batch_normalization_55 (Ba  (None, 512, 512, 64)      256       ['conv2d_58[0]
[0]']
 tchNormalization)

 activation_55 (Activation)  (None, 512, 512, 64)      0         ['batch_normal
ization_55[0][0]
                                                                  ']

 max_pooling2d_12 (MaxPooli  (None, 256, 256, 64)      0         ['activation_5
5[0][0]']
 ng2D)

 conv2d_59 (Conv2D)          (None, 256, 256, 128)     73856     ['max_pooling2
d_12[0][0]']

 batch_normalization_56 (Ba  (None, 256, 256, 128)     512       ['conv2d_59[0]
[0]']
 tchNormalization)

 activation_56 (Activation)  (None, 256, 256, 128)     0         ['batch_normal
ization_56[0][0]
                                                                  ']

 conv2d_60 (Conv2D)          (None, 256, 256, 128)     147584    ['activation_5
6[0][0]']

 batch_normalization_57 (Ba  (None, 256, 256, 128)     512       ['conv2d_60[0]
[0]']
 tchNormalization)

 activation_57 (Activation)  (None, 256, 256, 128)     0         ['batch_normal
ization_57[0][0]
                                                                  ']

 max_pooling2d_13 (MaxPooli  (None, 128, 128, 128)     0         ['activation_5
7[0][0]']
 ng2D)

 conv2d_61 (Conv2D)          (None, 128, 128, 256)     295168    ['max_pooling2
d_13[0][0]']

 batch_normalization_58 (Ba  (None, 128, 128, 256)     1024      ['conv2d_61[0]
```

```
[0]']
 tchNormalization)

 activation_58 (Activation)  (None, 128, 128, 256)        0         ['batch_normal
ization_58[0][0]
                                                                    ']

 conv2d_62 (Conv2D)          (None, 128, 128, 256)        590080    ['activation_5
8[0][0]']

 batch_normalization_59 (Ba  (None, 128, 128, 256)        1024      ['conv2d_62[0]
[0]']
 tchNormalization)

 activation_59 (Activation)  (None, 128, 128, 256)        0         ['batch_normal
ization_59[0][0]
                                                                    ']

 max_pooling2d_14 (MaxPooli  (None, 64, 64, 256)          0         ['activation_5
9[0][0]']
 ng2D)

 conv2d_63 (Conv2D)          (None, 64, 64, 512)          1180160   ['max_pooling2
d_14[0][0]']

 batch_normalization_60 (Ba  (None, 64, 64, 512)          2048      ['conv2d_63[0]
[0]']
 tchNormalization)

 activation_60 (Activation)  (None, 64, 64, 512)          0         ['batch_normal
ization_60[0][0]
                                                                    ']

 conv2d_64 (Conv2D)          (None, 64, 64, 512)          2359808   ['activation_6
0[0][0]']

 batch_normalization_61 (Ba  (None, 64, 64, 512)          2048      ['conv2d_64[0]
[0]']
 tchNormalization)

 activation_61 (Activation)  (None, 64, 64, 512)          0         ['batch_normal
ization_61[0][0]
                                                                    ']

 max_pooling2d_15 (MaxPooli  (None, 32, 32, 512)          0         ['activation_6
1[0][0]']
 ng2D)

 conv2d_65 (Conv2D)          (None, 32, 32, 1024)         4719616   ['max_pooling2
d_15[0][0]']

 batch_normalization_62 (Ba  (None, 32, 32, 1024)         4096      ['conv2d_65[0]
[0]']
 tchNormalization)

 activation_62 (Activation)  (None, 32, 32, 1024)         0         ['batch_normal
ization_62[0][0]
                                                                    ']

 conv2d_66 (Conv2D)          (None, 32, 32, 1024)         9438208   ['activation_6
2[0][0]']

 batch_normalization_63 (Ba  (None, 32, 32, 1024)         4096      ['conv2d_66[0]
[0]']
```

```
 tchNormalization)

 activation_63 (Activation)  (None, 32, 32, 1024)         0         ['batch_normal
ization_63[0][0]
                                                                    ']

 conv2d_transpose_12 (Conv2  (None, 64, 64, 512)          2097664   ['activation_6
3[0][0]']
 DTranspose)

 concatenate_12 (Concatenat  (None, 64, 64, 1024)         0         ['conv2d_trans
pose_12[0][0]',
 e)                                                                  'activation_6
1[0][0]']

 conv2d_67 (Conv2D)          (None, 64, 64, 512)          4719104   ['concatenate_
12[0][0]']

 batch_normalization_64 (Ba  (None, 64, 64, 512)          2048      ['conv2d_67[0]
[0]']
 tchNormalization)

 activation_64 (Activation)  (None, 64, 64, 512)          0         ['batch_normal
ization_64[0][0]
                                                                    ']

 conv2d_68 (Conv2D)          (None, 64, 64, 512)          2359808   ['activation_6
4[0][0]']

 batch_normalization_65 (Ba  (None, 64, 64, 512)          2048      ['conv2d_68[0]
[0]']
 tchNormalization)

 activation_65 (Activation)  (None, 64, 64, 512)          0         ['batch_normal
ization_65[0][0]
                                                                    ']

 conv2d_transpose_13 (Conv2  (None, 128, 128, 256)        524544    ['activation_6
5[0][0]']
 DTranspose)

 concatenate_13 (Concatenat  (None, 128, 128, 512)        0         ['conv2d_trans
pose_13[0][0]',
 e)                                                                  'activation_5
9[0][0]']

 conv2d_69 (Conv2D)          (None, 128, 128, 256)        1179904   ['concatenate_
13[0][0]']

 batch_normalization_66 (Ba  (None, 128, 128, 256)        1024      ['conv2d_69[0]
[0]']
 tchNormalization)

 activation_66 (Activation)  (None, 128, 128, 256)        0         ['batch_normal
ization_66[0][0]
                                                                    ']

 conv2d_70 (Conv2D)          (None, 128, 128, 256)        590080    ['activation_6
6[0][0]']

 batch_normalization_67 (Ba  (None, 128, 128, 256)        1024      ['conv2d_70[0]
[0]']
 tchNormalization)
```

```
 activation_67 (Activation)  (None, 128, 128, 256)        0         ['batch_normal
ization_67[0][0]
                                                                    ']

 conv2d_transpose_14 (Conv2  (None, 256, 256, 128)        131200    ['activation_6
7[0][0]']
 DTranspose)

 concatenate_14 (Concatenat  (None, 256, 256, 256)        0         ['conv2d_trans
pose_14[0][0]',
 e)                                                                  'activation_5
7[0][0]']

 conv2d_71 (Conv2D)          (None, 256, 256, 128)        295040    ['concatenate_
14[0][0]']

 batch_normalization_68 (Ba  (None, 256, 256, 128)        512       ['conv2d_71[0]
[0]']
 tchNormalization)

 activation_68 (Activation)  (None, 256, 256, 128)        0         ['batch_normal
ization_68[0][0]
                                                                    ']

 conv2d_72 (Conv2D)          (None, 256, 256, 128)        147584    ['activation_6
8[0][0]']

 batch_normalization_69 (Ba  (None, 256, 256, 128)        512       ['conv2d_72[0]
[0]']
 tchNormalization)

 activation_69 (Activation)  (None, 256, 256, 128)        0         ['batch_normal
ization_69[0][0]
                                                                    ']

 conv2d_transpose_15 (Conv2  (None, 512, 512, 64)         32832     ['activation_6
9[0][0]']
 DTranspose)

 concatenate_15 (Concatenat  (None, 512, 512, 128)        0         ['conv2d_trans
pose_15[0][0]',
 e)                                                                  'activation_5
5[0][0]']

 conv2d_73 (Conv2D)          (None, 512, 512, 64)         73792     ['concatenate_
15[0][0]']

 batch_normalization_70 (Ba  (None, 512, 512, 64)         256       ['conv2d_73[0]
[0]']
 tchNormalization)

 activation_70 (Activation)  (None, 512, 512, 64)         0         ['batch_normal
ization_70[0][0]
                                                                    ']

 conv2d_74 (Conv2D)          (None, 512, 512, 64)         36928     ['activation_7
0[0][0]']

 batch_normalization_71 (Ba  (None, 512, 512, 64)         256       ['conv2d_74[0]
[0]']
 tchNormalization)

 activation_71 (Activation)  (None, 512, 512, 64)         0         ['batch_normal
ization_71[0][0]
```

```
                                                                                ']

 conv2d_75 (Conv2D)          (None, 512, 512, 1)          65        ['activation_7
1[0][0]']

==================================================================================
================
Total params: 31055297 (118.47 MB)
Trainable params: 31043521 (118.42 MB)
Non-trainable params: 11776 (46.00 KB)
__________________________________________________________________________________
________________
```

In [45]:
```
unet_result = unet_model.fit(
    images_train, masks_train,
    validation_data =(images_test, masks_test), batch_size = 4, epochs =100)
```

```
Epoch 1/100
6/6 [==============================] - 27s 3s/step - loss: -47.7629 - accuracy: 0.
2348 - val_loss: -145.2551 - val_accuracy: 6.8283e-04
Epoch 2/100
6/6 [==============================] - 6s 1s/step - loss: -95.7591 - accuracy: 0.3
995 - val_loss: -251.8573 - val_accuracy: 5.6839e-04
Epoch 3/100
6/6 [==============================] - 6s 1s/step - loss: -138.6153 - accuracy: 0.
4692 - val_loss: -337.8133 - val_accuracy: 0.0040
Epoch 4/100
6/6 [==============================] - 6s 1s/step - loss: -181.9978 - accuracy: 0.
5129 - val_loss: -421.7184 - val_accuracy: 0.0113
Epoch 5/100
6/6 [==============================] - 6s 1s/step - loss: -219.8105 - accuracy: 0.
5525 - val_loss: -501.1974 - val_accuracy: 0.0157
Epoch 6/100
6/6 [==============================] - 6s 1s/step - loss: -241.6008 - accuracy: 0.
5854 - val_loss: -574.8091 - val_accuracy: 0.0210
Epoch 7/100
6/6 [==============================] - 6s 1s/step - loss: -268.9613 - accuracy: 0.
6244 - val_loss: -513.0665 - val_accuracy: 0.0246
Epoch 8/100
6/6 [==============================] - 6s 1s/step - loss: -295.2440 - accuracy: 0.
6484 - val_loss: -512.3088 - val_accuracy: 0.0328
Epoch 9/100
6/6 [==============================] - 6s 1s/step - loss: -307.3561 - accuracy: 0.
6592 - val_loss: -490.8855 - val_accuracy: 0.0550
Epoch 10/100
6/6 [==============================] - 6s 1s/step - loss: -330.6661 - accuracy: 0.
6804 - val_loss: -426.9705 - val_accuracy: 0.1021
Epoch 11/100
6/6 [==============================] - 6s 1s/step - loss: -341.2258 - accuracy: 0.
6904 - val_loss: -355.5688 - val_accuracy: 0.1756
Epoch 12/100
6/6 [==============================] - 6s 1s/step - loss: -350.6190 - accuracy: 0.
6923 - val_loss: -321.1351 - val_accuracy: 0.2513
Epoch 13/100
6/6 [==============================] - 6s 1s/step - loss: -368.8701 - accuracy: 0.
7048 - val_loss: -263.8867 - val_accuracy: 0.3393
Epoch 14/100
6/6 [==============================] - 6s 1s/step - loss: -378.0547 - accuracy: 0.
7170 - val_loss: -215.1730 - val_accuracy: 0.4255
Epoch 15/100
6/6 [==============================] - 6s 1s/step - loss: -393.5399 - accuracy: 0.
7271 - val_loss: -194.0864 - val_accuracy: 0.4797
Epoch 16/100
6/6 [==============================] - 6s 1s/step - loss: -399.9136 - accuracy: 0.
7285 - val_loss: -183.6939 - val_accuracy: 0.5295
Epoch 17/100
6/6 [==============================] - 6s 1s/step - loss: -407.0480 - accuracy: 0.
7302 - val_loss: -135.8284 - val_accuracy: 0.6036
Epoch 18/100
6/6 [==============================] - 6s 1s/step - loss: -416.7238 - accuracy: 0.
7431 - val_loss: -126.5182 - val_accuracy: 0.6471
Epoch 19/100
6/6 [==============================] - 6s 1s/step - loss: -420.2852 - accuracy: 0.
7367 - val_loss: -118.1956 - val_accuracy: 0.6813
Epoch 20/100
6/6 [==============================] - 6s 1s/step - loss: -433.7156 - accuracy: 0.
7512 - val_loss: -118.8167 - val_accuracy: 0.7025
Epoch 21/100
6/6 [==============================] - 6s 1s/step - loss: -437.3686 - accuracy: 0.
7451 - val_loss: -116.5912 - val_accuracy: 0.7246
Epoch 22/100
```

```
6/6 [==============================] - 6s 1s/step - loss: -444.4846 - accuracy: 0.
7539 - val_loss: -98.9792 - val_accuracy: 0.7523
Epoch 23/100
6/6 [==============================] - 6s 1s/step - loss: -452.1189 - accuracy: 0.
7581 - val_loss: -113.2442 - val_accuracy: 0.7550
Epoch 24/100
6/6 [==============================] - 6s 1s/step - loss: -455.3216 - accuracy: 0.
7604 - val_loss: -97.7468 - val_accuracy: 0.7767
Epoch 25/100
6/6 [==============================] - 6s 1s/step - loss: -455.7027 - accuracy: 0.
7598 - val_loss: -104.6411 - val_accuracy: 0.7799
Epoch 26/100
6/6 [==============================] - 6s 1s/step - loss: -466.7805 - accuracy: 0.
7653 - val_loss: -109.3446 - val_accuracy: 0.7836
Epoch 27/100
6/6 [==============================] - 6s 1s/step - loss: -467.9897 - accuracy: 0.
7594 - val_loss: -101.0304 - val_accuracy: 0.7910
Epoch 28/100
6/6 [==============================] - 6s 1s/step - loss: -473.5247 - accuracy: 0.
7646 - val_loss: -114.4826 - val_accuracy: 0.7887
Epoch 29/100
6/6 [==============================] - 6s 1s/step - loss: -481.5355 - accuracy: 0.
7693 - val_loss: -116.9257 - val_accuracy: 0.7912
Epoch 30/100
6/6 [==============================] - 6s 1s/step - loss: -482.7085 - accuracy: 0.
7680 - val_loss: -126.9114 - val_accuracy: 0.7906
Epoch 31/100
6/6 [==============================] - 6s 1s/step - loss: -486.0682 - accuracy: 0.
7683 - val_loss: -121.3913 - val_accuracy: 0.7939
Epoch 32/100
6/6 [==============================] - 6s 1s/step - loss: -495.1865 - accuracy: 0.
7726 - val_loss: -155.1726 - val_accuracy: 0.7858
Epoch 33/100
6/6 [==============================] - 6s 1s/step - loss: -495.5182 - accuracy: 0.
7717 - val_loss: -148.1943 - val_accuracy: 0.7904
Epoch 34/100
6/6 [==============================] - 6s 1s/step - loss: -497.1404 - accuracy: 0.
7745 - val_loss: -128.1922 - val_accuracy: 0.7960
Epoch 35/100
6/6 [==============================] - 6s 1s/step - loss: -503.0397 - accuracy: 0.
7730 - val_loss: -131.5654 - val_accuracy: 0.7973
Epoch 36/100
6/6 [==============================] - 6s 1s/step - loss: -511.2599 - accuracy: 0.
7770 - val_loss: -177.3412 - val_accuracy: 0.7866
Epoch 37/100
6/6 [==============================] - 6s 1s/step - loss: -511.5766 - accuracy: 0.
7783 - val_loss: -192.0101 - val_accuracy: 0.7845
Epoch 38/100
6/6 [==============================] - 6s 1s/step - loss: -519.0371 - accuracy: 0.
7757 - val_loss: -150.2967 - val_accuracy: 0.7949
Epoch 39/100
6/6 [==============================] - 6s 1s/step - loss: -513.9775 - accuracy: 0.
7731 - val_loss: -204.2843 - val_accuracy: 0.7841
Epoch 40/100
6/6 [==============================] - 6s 1s/step - loss: -521.7351 - accuracy: 0.
7818 - val_loss: -252.4574 - val_accuracy: 0.7719
Epoch 41/100
6/6 [==============================] - 6s 1s/step - loss: -521.7729 - accuracy: 0.
7706 - val_loss: -191.9733 - val_accuracy: 0.7886
Epoch 42/100
6/6 [==============================] - 6s 1s/step - loss: -530.8376 - accuracy: 0.
7810 - val_loss: -256.7462 - val_accuracy: 0.7745
Epoch 43/100
6/6 [==============================] - 6s 1s/step - loss: -522.6811 - accuracy: 0.
```

```
7706 - val_loss: -237.3082 - val_accuracy: 0.7809
Epoch 44/100
6/6 [==============================] - 6s 1s/step - loss: -525.9550 - accuracy: 0.
7834 - val_loss: -282.6603 - val_accuracy: 0.7725
Epoch 45/100
6/6 [==============================] - 6s 1s/step - loss: -541.1309 - accuracy: 0.
7785 - val_loss: -228.9198 - val_accuracy: 0.7819
Epoch 46/100
6/6 [==============================] - 6s 1s/step - loss: -536.2256 - accuracy: 0.
7803 - val_loss: -328.3146 - val_accuracy: 0.7629
Epoch 47/100
6/6 [==============================] - 6s 1s/step - loss: -541.4224 - accuracy: 0.
7812 - val_loss: -265.7115 - val_accuracy: 0.7788
Epoch 48/100
6/6 [==============================] - 6s 1s/step - loss: -545.6782 - accuracy: 0.
7801 - val_loss: -238.5154 - val_accuracy: 0.7862
Epoch 49/100
6/6 [==============================] - 6s 1s/step - loss: -552.4739 - accuracy: 0.
7822 - val_loss: -304.4744 - val_accuracy: 0.7715
Epoch 50/100
6/6 [==============================] - 6s 1s/step - loss: -560.3397 - accuracy: 0.
7849 - val_loss: -301.4961 - val_accuracy: 0.7732
Epoch 51/100
6/6 [==============================] - 6s 1s/step - loss: -557.5082 - accuracy: 0.
7815 - val_loss: -302.5273 - val_accuracy: 0.7712
Epoch 52/100
6/6 [==============================] - 6s 1s/step - loss: -563.3322 - accuracy: 0.
7821 - val_loss: -319.5435 - val_accuracy: 0.7681
Epoch 53/100
6/6 [==============================] - 6s 1s/step - loss: -568.0286 - accuracy: 0.
7843 - val_loss: -350.0312 - val_accuracy: 0.7623
Epoch 54/100
6/6 [==============================] - 6s 1s/step - loss: -572.2701 - accuracy: 0.
7862 - val_loss: -311.8686 - val_accuracy: 0.7718
Epoch 55/100
6/6 [==============================] - 6s 1s/step - loss: -577.2328 - accuracy: 0.
7852 - val_loss: -318.4034 - val_accuracy: 0.7698
Epoch 56/100
6/6 [==============================] - 6s 1s/step - loss: -576.5402 - accuracy: 0.
7870 - val_loss: -356.7955 - val_accuracy: 0.7606
Epoch 57/100
6/6 [==============================] - 6s 1s/step - loss: -582.6329 - accuracy: 0.
7843 - val_loss: -350.9704 - val_accuracy: 0.7638
Epoch 58/100
6/6 [==============================] - 6s 1s/step - loss: -585.9571 - accuracy: 0.
7886 - val_loss: -344.1150 - val_accuracy: 0.7648
Epoch 59/100
6/6 [==============================] - 6s 1s/step - loss: -587.5783 - accuracy: 0.
7864 - val_loss: -343.7081 - val_accuracy: 0.7666
Epoch 60/100
6/6 [==============================] - 6s 1s/step - loss: -588.7645 - accuracy: 0.
7838 - val_loss: -311.8725 - val_accuracy: 0.7770
Epoch 61/100
6/6 [==============================] - 6s 1s/step - loss: -597.8285 - accuracy: 0.
7898 - val_loss: -370.9767 - val_accuracy: 0.7598
Epoch 62/100
6/6 [==============================] - 6s 1s/step - loss: -595.4824 - accuracy: 0.
7841 - val_loss: -341.8932 - val_accuracy: 0.7630
Epoch 63/100
6/6 [==============================] - 6s 1s/step - loss: -596.4656 - accuracy: 0.
7845 - val_loss: -447.7199 - val_accuracy: 0.7394
Epoch 64/100
6/6 [==============================] - 6s 1s/step - loss: -600.6525 - accuracy: 0.
7884 - val_loss: -399.3064 - val_accuracy: 0.7409
```

```
Epoch 65/100
6/6 [==============================] - 6s 1s/step - loss: -601.6725 - accuracy: 0.
7887 - val_loss: -349.7931 - val_accuracy: 0.7635
Epoch 66/100
6/6 [==============================] - 6s 1s/step - loss: -613.6378 - accuracy: 0.
7869 - val_loss: -374.4945 - val_accuracy: 0.7619
Epoch 67/100
6/6 [==============================] - 6s 1s/step - loss: -615.0887 - accuracy: 0.
7847 - val_loss: -411.6403 - val_accuracy: 0.7526
Epoch 68/100
6/6 [==============================] - 6s 1s/step - loss: -610.5045 - accuracy: 0.
7906 - val_loss: -362.3378 - val_accuracy: 0.7595
Epoch 69/100
6/6 [==============================] - 6s 1s/step - loss: -613.7285 - accuracy: 0.
7876 - val_loss: -404.6571 - val_accuracy: 0.7500
Epoch 70/100
6/6 [==============================] - 6s 1s/step - loss: -621.5555 - accuracy: 0.
7862 - val_loss: -382.2899 - val_accuracy: 0.7612
Epoch 71/100
6/6 [==============================] - 6s 1s/step - loss: -624.7530 - accuracy: 0.
7855 - val_loss: -318.0841 - val_accuracy: 0.7790
Epoch 72/100
6/6 [==============================] - 6s 1s/step - loss: -626.4772 - accuracy: 0.
7894 - val_loss: -347.3641 - val_accuracy: 0.7652
Epoch 73/100
6/6 [==============================] - 6s 1s/step - loss: -631.7681 - accuracy: 0.
7892 - val_loss: -464.5752 - val_accuracy: 0.7315
Epoch 74/100
6/6 [==============================] - 6s 1s/step - loss: -635.1624 - accuracy: 0.
7881 - val_loss: -391.5944 - val_accuracy: 0.7547
Epoch 75/100
6/6 [==============================] - 6s 1s/step - loss: -640.5854 - accuracy: 0.
7895 - val_loss: -382.1852 - val_accuracy: 0.7576
Epoch 76/100
6/6 [==============================] - 6s 1s/step - loss: -633.3234 - accuracy: 0.
7883 - val_loss: -383.0846 - val_accuracy: 0.7615
Epoch 77/100
6/6 [==============================] - 6s 1s/step - loss: -646.9368 - accuracy: 0.
7909 - val_loss: -367.1524 - val_accuracy: 0.7629
Epoch 78/100
6/6 [==============================] - 6s 1s/step - loss: -653.4293 - accuracy: 0.
7908 - val_loss: -483.2713 - val_accuracy: 0.7342
Epoch 79/100
6/6 [==============================] - 6s 1s/step - loss: -655.7564 - accuracy: 0.
7906 - val_loss: -455.4154 - val_accuracy: 0.7259
Epoch 80/100
6/6 [==============================] - 6s 1s/step - loss: -658.4677 - accuracy: 0.
7905 - val_loss: -421.3471 - val_accuracy: 0.7473
Epoch 81/100
6/6 [==============================] - 6s 1s/step - loss: -661.9708 - accuracy: 0.
7907 - val_loss: -460.3384 - val_accuracy: 0.7286
Epoch 82/100
6/6 [==============================] - 6s 1s/step - loss: -657.6202 - accuracy: 0.
7873 - val_loss: -522.4274 - val_accuracy: 0.7049
Epoch 83/100
6/6 [==============================] - 6s 1s/step - loss: -666.6270 - accuracy: 0.
7900 - val_loss: -438.1235 - val_accuracy: 0.7481
Epoch 84/100
6/6 [==============================] - 6s 1s/step - loss: -671.4764 - accuracy: 0.
7911 - val_loss: -437.1808 - val_accuracy: 0.7320
Epoch 85/100
6/6 [==============================] - 6s 1s/step - loss: -667.7960 - accuracy: 0.
7889 - val_loss: -468.5204 - val_accuracy: 0.7286
Epoch 86/100
```

```
6/6 [==============================] - 6s 1s/step - loss: -678.3326 - accuracy: 0.
7932 - val_loss: -493.6963 - val_accuracy: 0.7176
Epoch 87/100
6/6 [==============================] - 6s 1s/step - loss: -680.9562 - accuracy: 0.
7905 - val_loss: -436.6368 - val_accuracy: 0.7385
Epoch 88/100
6/6 [==============================] - 6s 1s/step - loss: -683.1550 - accuracy: 0.
7911 - val_loss: -413.5240 - val_accuracy: 0.7471
Epoch 89/100
6/6 [==============================] - 6s 1s/step - loss: -676.3268 - accuracy: 0.
7822 - val_loss: -635.0498 - val_accuracy: 0.6681
Epoch 90/100
6/6 [==============================] - 6s 1s/step - loss: -685.8035 - accuracy: 0.
7896 - val_loss: -455.0648 - val_accuracy: 0.7389
Epoch 91/100
6/6 [==============================] - 6s 1s/step - loss: -691.2653 - accuracy: 0.
7927 - val_loss: -440.3832 - val_accuracy: 0.7509
Epoch 92/100
6/6 [==============================] - 6s 1s/step - loss: -694.9678 - accuracy: 0.
7940 - val_loss: -485.0518 - val_accuracy: 0.7263
Epoch 93/100
6/6 [==============================] - 6s 1s/step - loss: -699.1989 - accuracy: 0.
7923 - val_loss: -450.6559 - val_accuracy: 0.7408
Epoch 94/100
6/6 [==============================] - 6s 1s/step - loss: -702.9719 - accuracy: 0.
7952 - val_loss: -521.3723 - val_accuracy: 0.7121
Epoch 95/100
6/6 [==============================] - 6s 1s/step - loss: -692.0285 - accuracy: 0.
7921 - val_loss: -436.6462 - val_accuracy: 0.7476
Epoch 96/100
6/6 [==============================] - 6s 1s/step - loss: -700.9548 - accuracy: 0.
7938 - val_loss: -402.3509 - val_accuracy: 0.7635
Epoch 97/100
6/6 [==============================] - 6s 1s/step - loss: -705.4036 - accuracy: 0.
7919 - val_loss: -466.9543 - val_accuracy: 0.7481
Epoch 98/100
6/6 [==============================] - 6s 1s/step - loss: -714.3970 - accuracy: 0.
7939 - val_loss: -483.3735 - val_accuracy: 0.7342
Epoch 99/100
6/6 [==============================] - 6s 1s/step - loss: -711.8196 - accuracy: 0.
7963 - val_loss: -456.8835 - val_accuracy: 0.7324
Epoch 100/100
6/6 [==============================] - 6s 1s/step - loss: -723.0023 - accuracy: 0.
7932 - val_loss: -500.5265 - val_accuracy: 0.7399
```

In [46]:
```
def show_result(idx, og, unet, target, p):

    fig, axs = plt.subplots(1, 3, figsize=(12,12))
    axs[0].set_title("Original "+str(idx) )
    axs[0].imshow(og)
    axs[0].axis('off')

    axs[1].set_title("U-Net: p>"+str(p))
    axs[1].imshow(unet)
    axs[1].axis('off')

    axs[2].set_title("Ground Truth")
    axs[2].imshow(target)
    axs[2].axis('off')

    plt.show()
```

In [47]:
```
unet_predict = unet_model.predict(images_test)
```

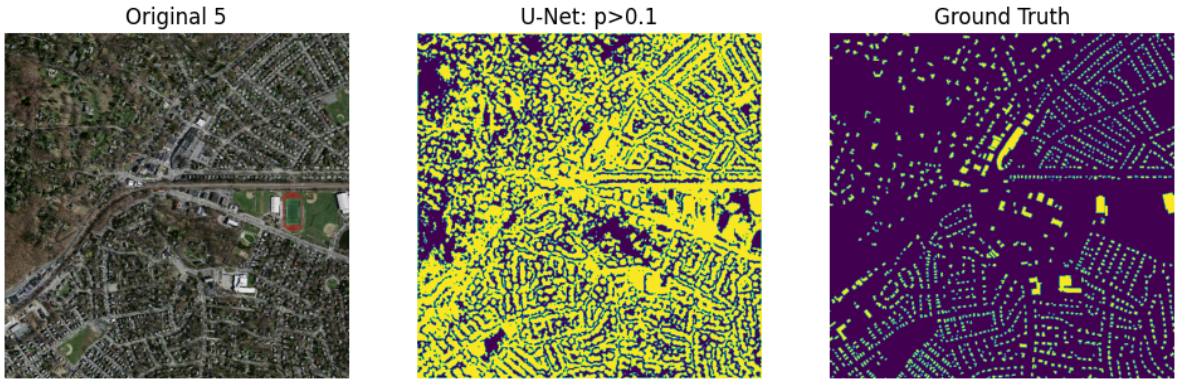
```
1/1 [==============================] - 20s 20s/step
```

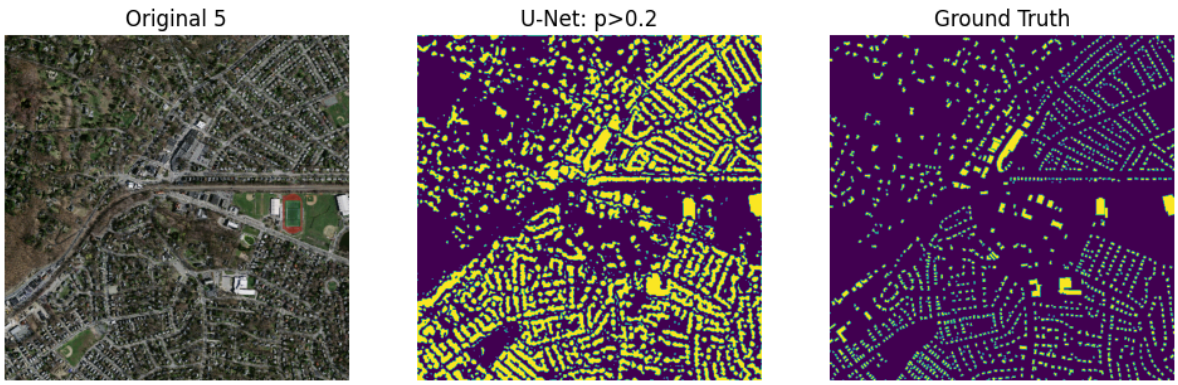
In [48]:
```
len(images_test)
```

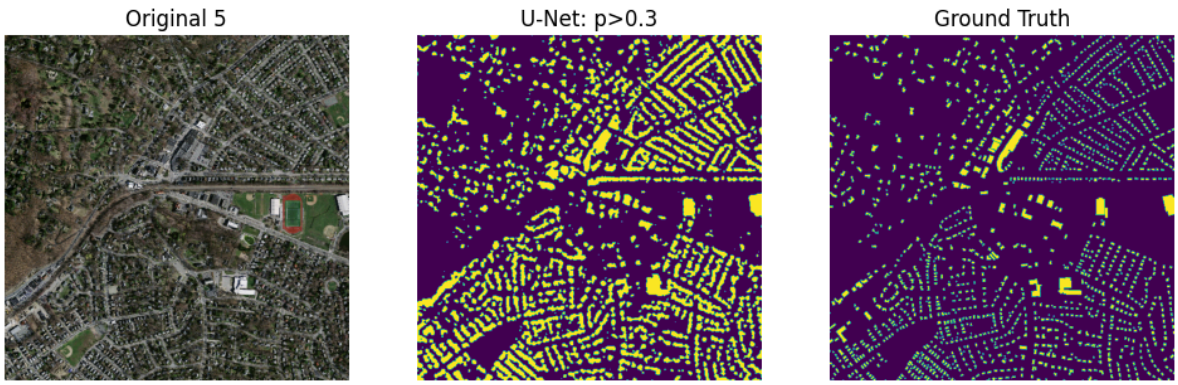
Out[48]:
```
6
```

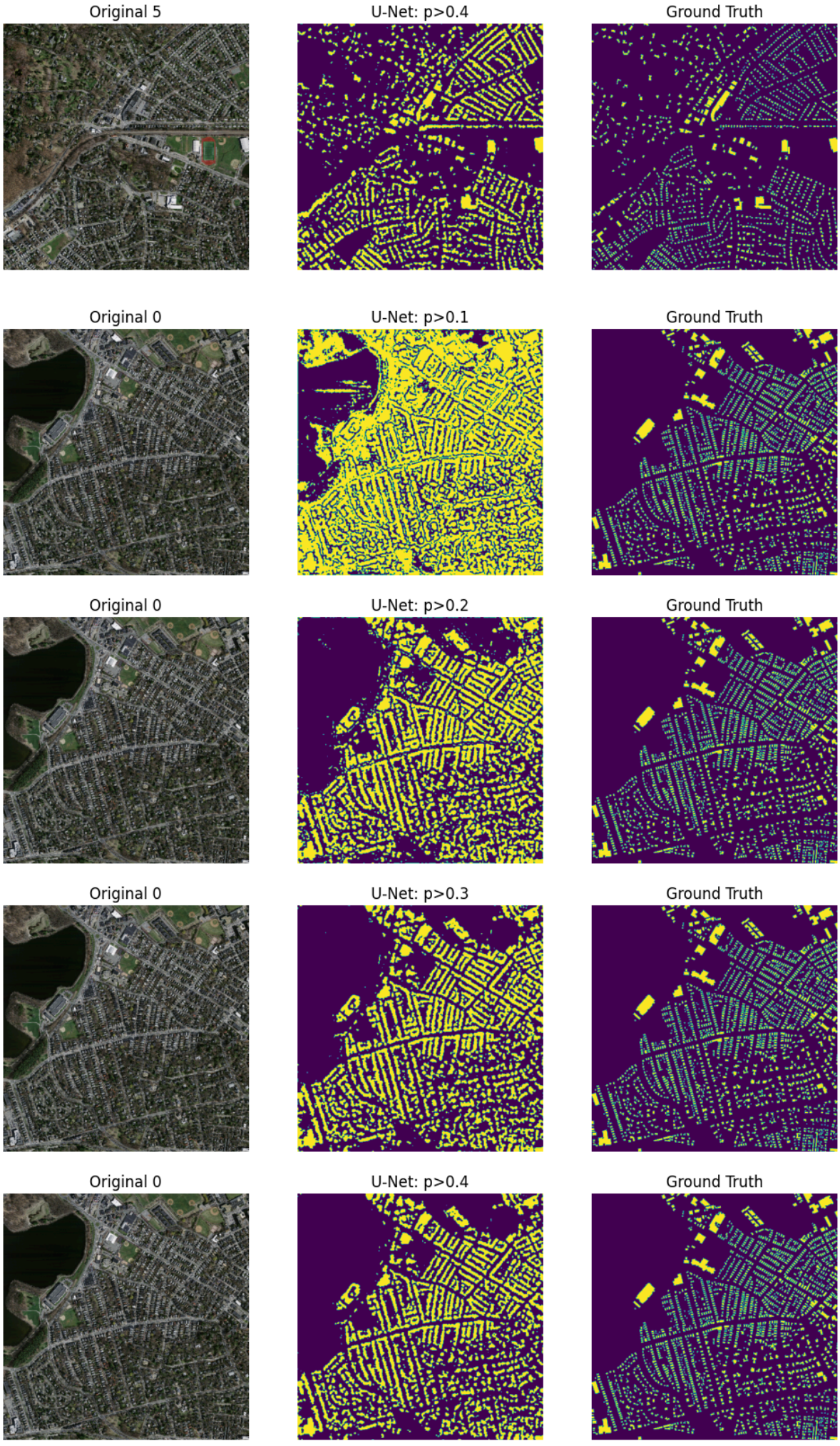
In [49]:
```
unet_predict1 = (unet_predict > 0.1).astype(np.uint8)
unet_predict2 = (unet_predict > 0.2).astype(np.uint8)
unet_predict3 = (unet_predict > 0.3).astype(np.uint8)
unet_predict4 = (unet_predict > 0.4).astype(np.uint8)
```

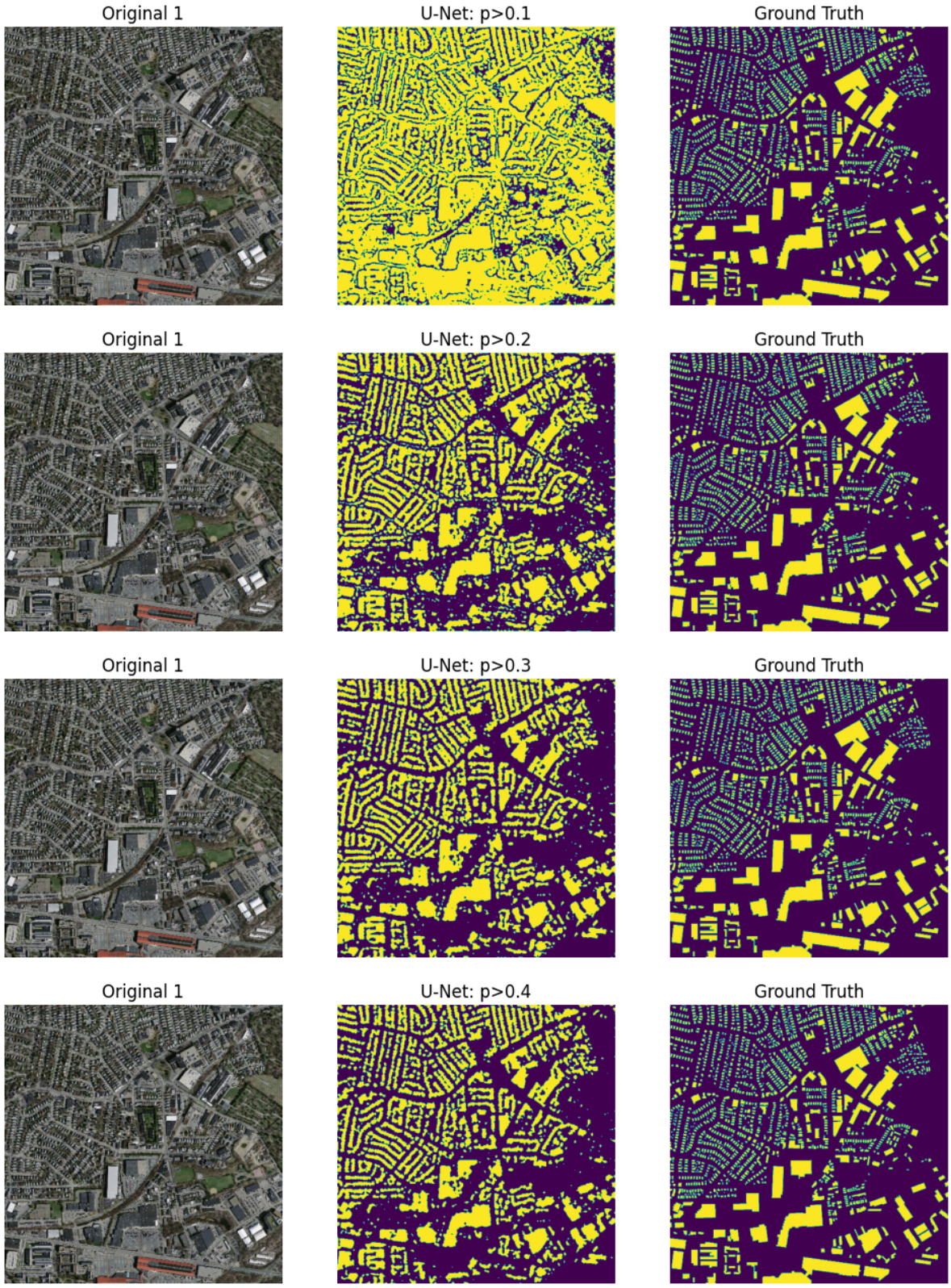
In [50]:
```
show_test_idx = random.sample(range(len(unet_predict)), 3)
for idx in show_test_idx:
    show_result(idx, images_test[idx], unet_predict1[idx], masks_test[idx], 0.1)
    show_result(idx, images_test[idx], unet_predict2[idx], masks_test[idx], 0.2)
    show_result(idx, images_test[idx], unet_predict3[idx], masks_test[idx], 0.3)
    show_result(idx, images_test[idx], unet_predict4[idx], masks_test[idx], 0.4)
    print()
```








Original 5
U-Net: p>0.4
Ground Truth
Original 0
U-Net: p>0.1
Ground Truth
Original 0
U-Net: p>0.2
Ground Truth
Original 0
U-Net: p>0.3
Ground Truth
Original 0
U-Net: p>0.4
Ground Truth



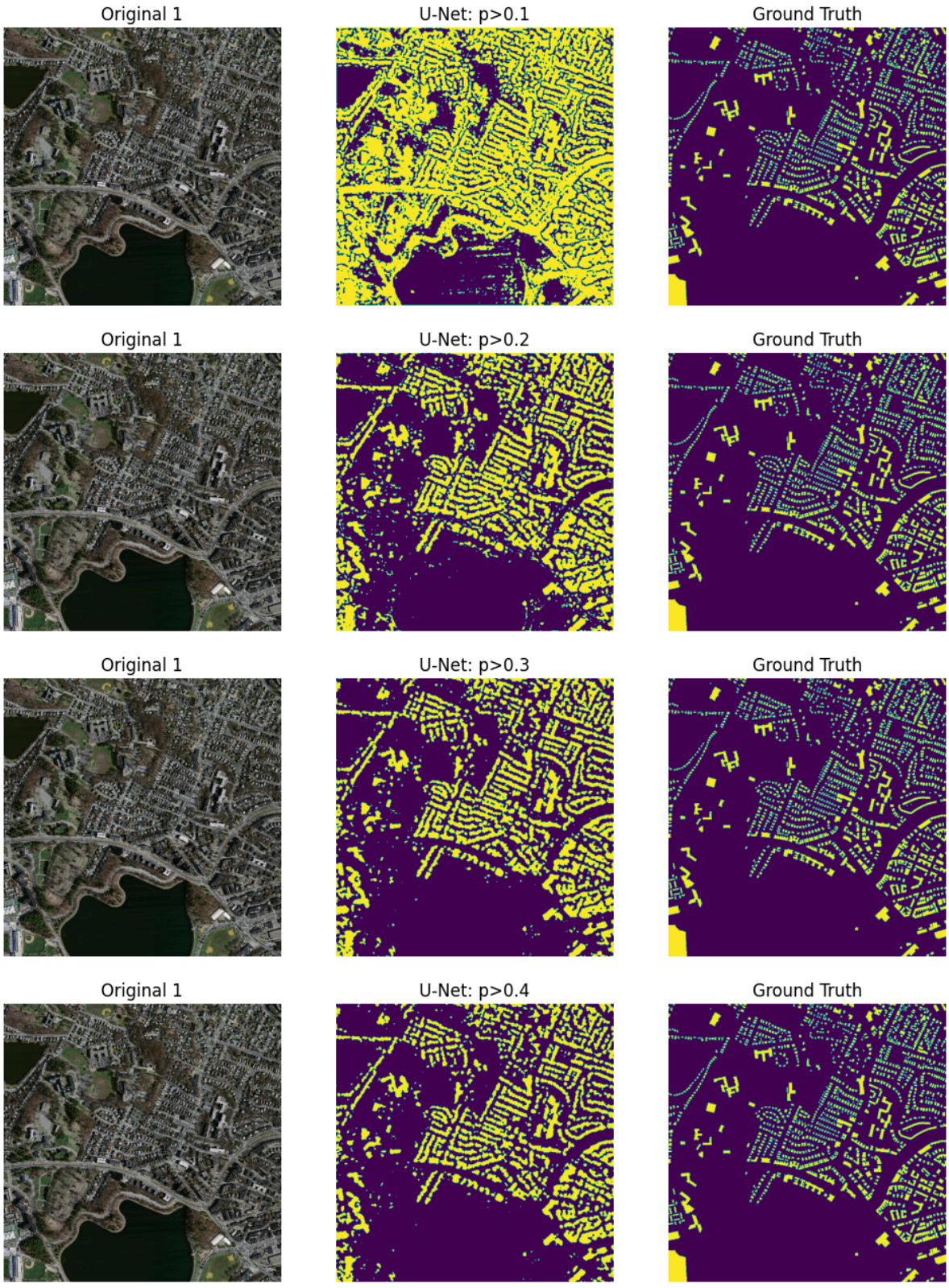
In [51]:
```
unet_predictT = unet_model.predict(test_images)
```

```
1/1 [==============================] - 0s 27ms/step
```

In [52]:
```
unet_predict1 = (unet_predictT > 0.1).astype(np.uint8)
unet_predict2 = (unet_predictT > 0.2).astype(np.uint8)
unet_predict3 = (unet_predictT > 0.3).astype(np.uint8)
unet_predict4 = (unet_predictT > 0.4).astype(np.uint8)
```

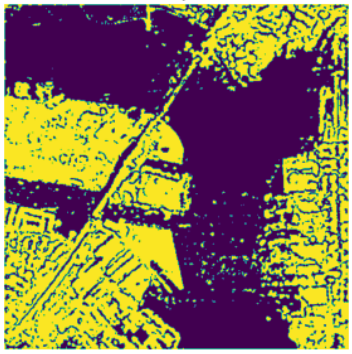
In [53]:
```
show_test_idx = random.sample(range(len(unet_predictT)), 3)
for idx in show_test_idx:
    show_result(idx, test_images[idx], unet_predict1[idx], test_masks[idx], 0.1)
```

```
    show_result(idx, test_images[idx], unet_predict2[idx], test_masks[idx], 0.2)
    show_result(idx, test_images[idx], unet_predict3[idx], test_masks[idx], 0.3)
    show_result(idx, test_images[idx], unet_predict4[idx], test_masks[idx], 0.4)
    print()
```

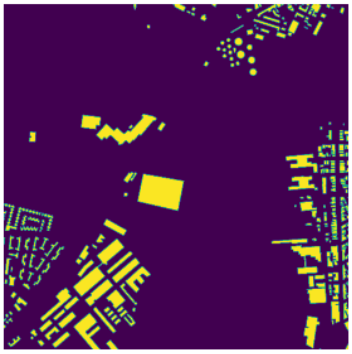


Original 7

U-Net: p>0.1



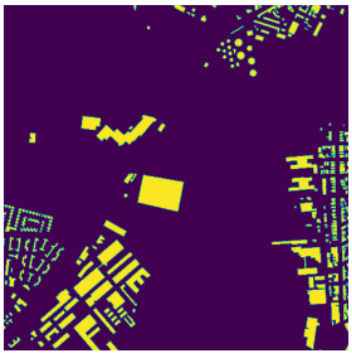
Ground Truth



Original 7

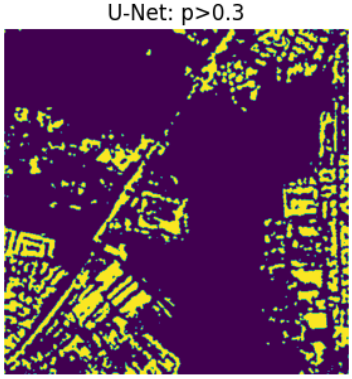
U-Net: p>0.2

Ground Truth

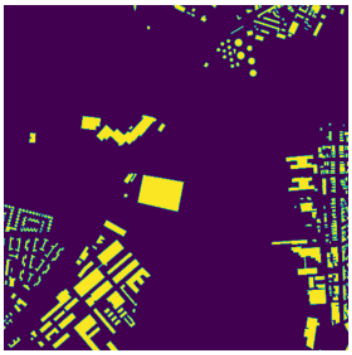


Original 7

U-Net: p>0.3

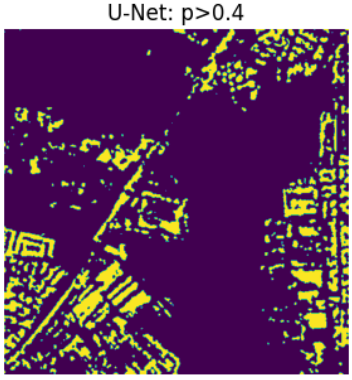


Ground Truth

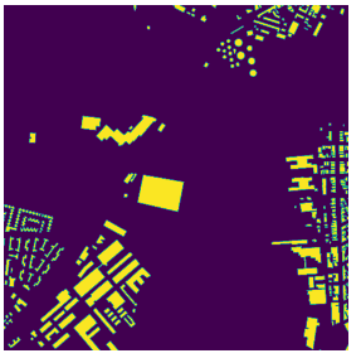


Original 7

U-Net: p>0.4

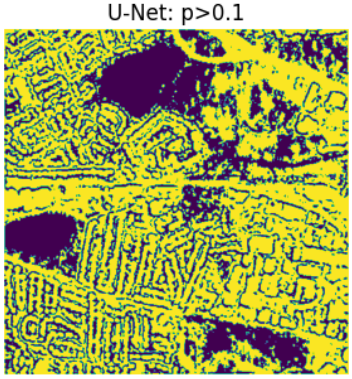


Ground Truth

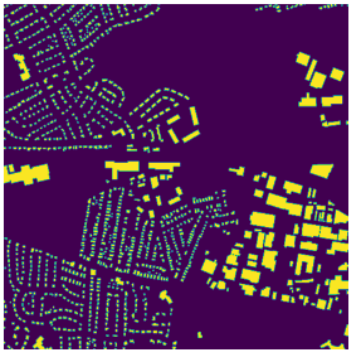


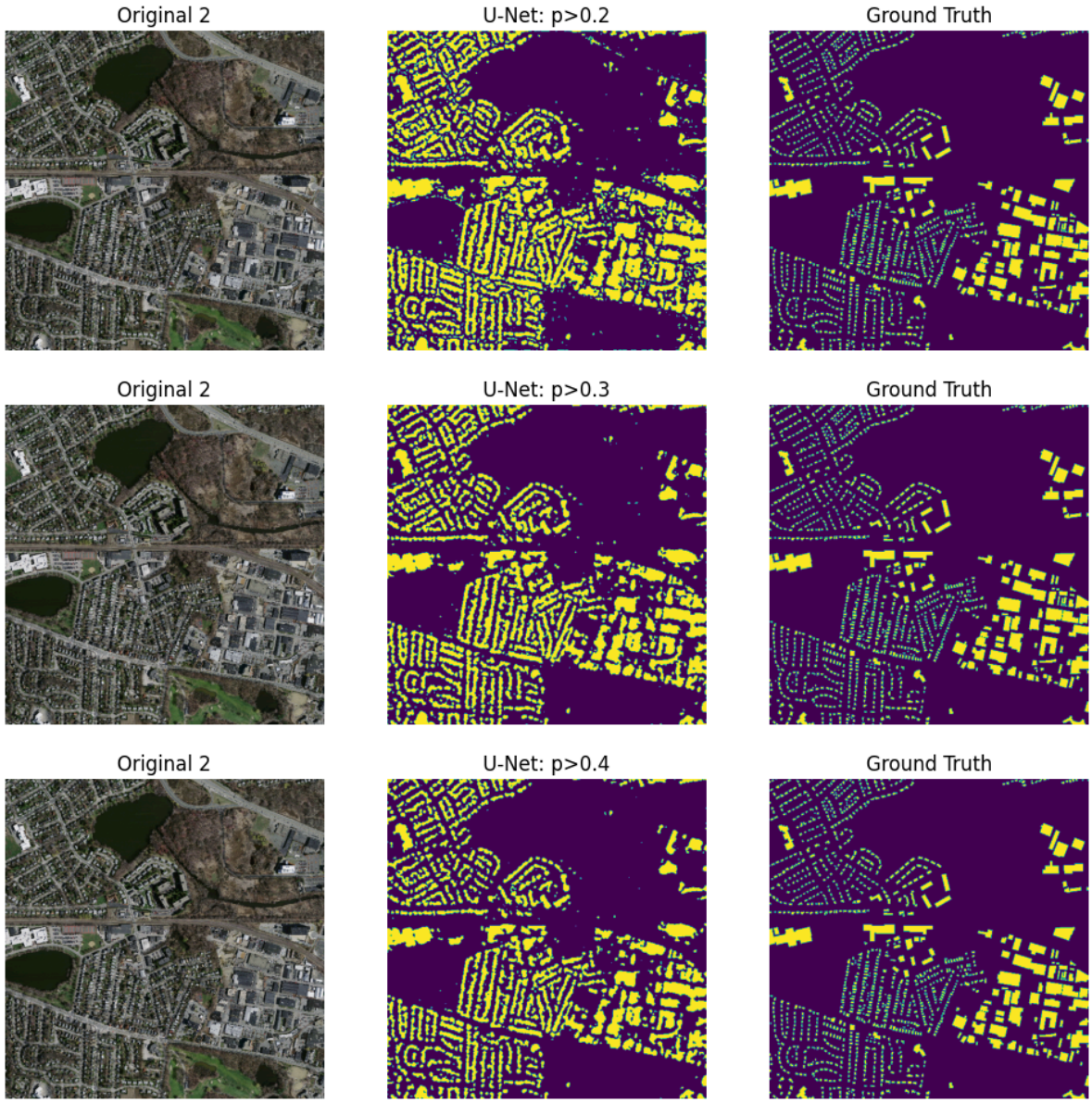
Original 2

U-Net: p>0.1



Ground Truth




Original 2
U-Net: p>0.2
Ground Truth
Original 2
U-Net: p>0.3
Ground Truth
Original 2
U-Net: p>0.4
Ground Truth

In [ ]: