In [2]:
```
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')
```

```
Mounted at /content/drive
```

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
```

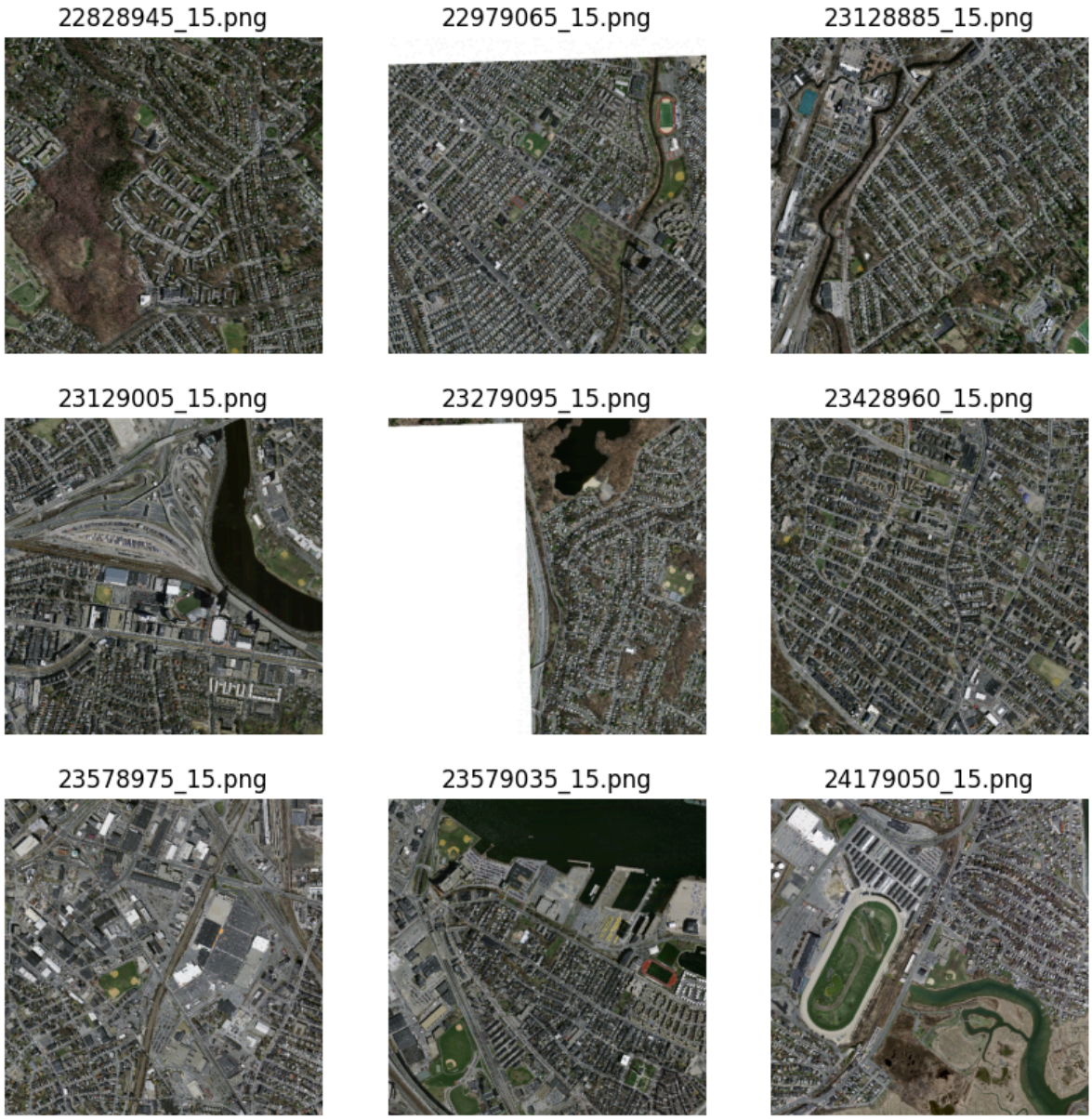
In [7]:
```
image_size=512
input_image_size=(512,512)
```

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
```

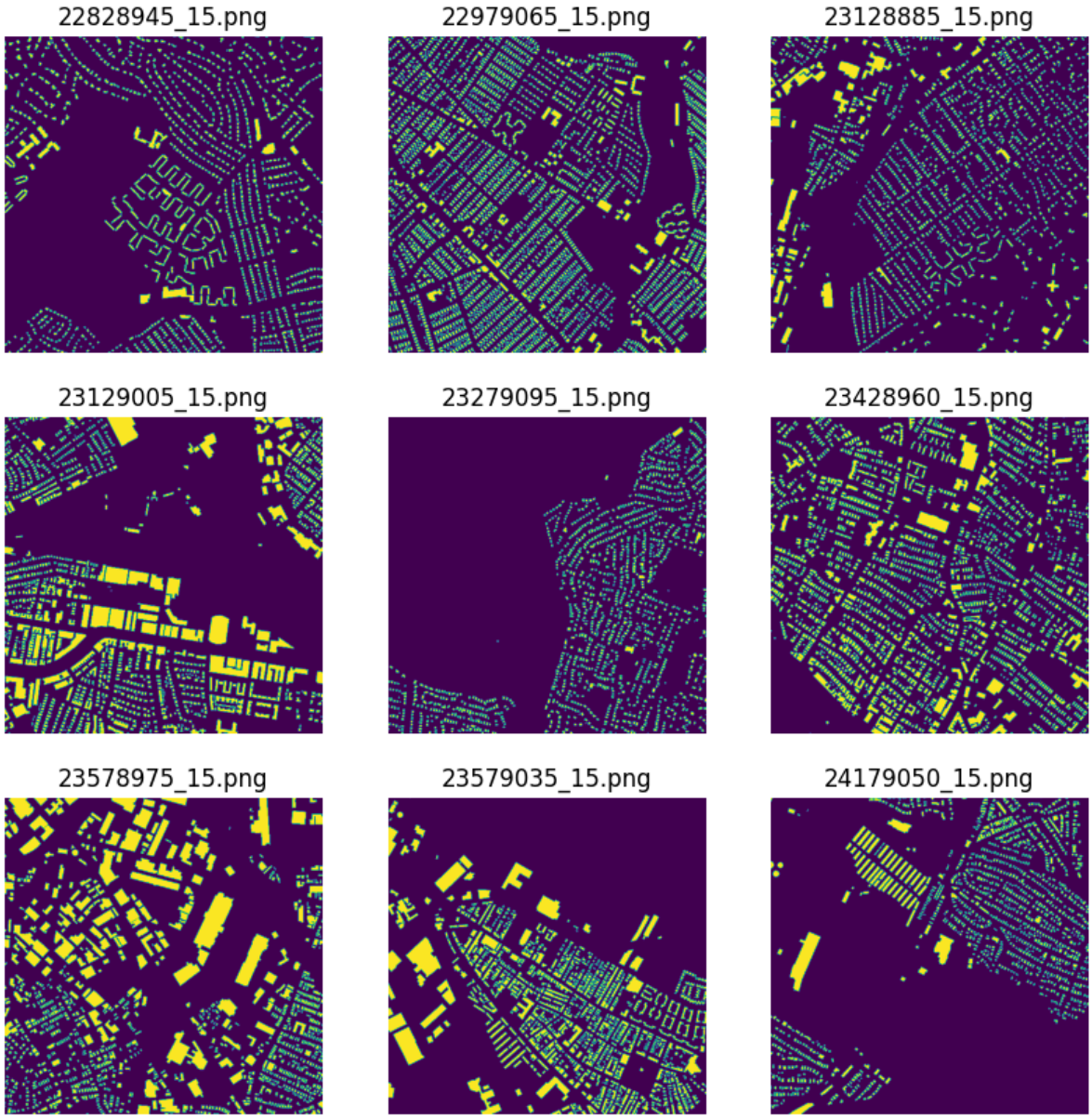
In [9]:
```
number = 30
```

In [10]:
```
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 [11]:
```
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 [12]:
```
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 [13]:
```
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 [14]:
```
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 [15]:
```
images_train, images_test, masks_train, masks_test = train_test_split(images, masks
print(len(images_train), len(masks_train))
```

```
24 24
```

In [16]:
```
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=0.00001), loss= 'binary_crossentrop
unet_model.summary()
```

```
Model: "U-Net"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 512, 512, 3)]     0         []

 conv2d (Conv2D)             (None, 512, 512, 64)      1792      ['input_1[0]
[0]']

 batch_normalization (Batch  (None, 512, 512, 64)      256       ['conv2d[0]
[0]']
 Normalization)

 activation (Activation)     (None, 512, 512, 64)      0         ['batch_normal
ization[0][0]']

 conv2d_1 (Conv2D)           (None, 512, 512, 64)      36928     ['activation
[0][0]']

 batch_normalization_1 (Bat  (None, 512, 512, 64)      256       ['conv2d_1[0]
[0]']
 chNormalization)

 activation_1 (Activation)   (None, 512, 512, 64)      0         ['batch_normal
ization_1[0][0]'
                                                                 ]

 max_pooling2d (MaxPooling2  (None, 256, 256, 64)      0         ['activation_1
[0][0]']
 D)

 conv2d_2 (Conv2D)           (None, 256, 256, 128)     73856     ['max_pooling2
d[0][0]']

 batch_normalization_2 (Bat  (None, 256, 256, 128)     512       ['conv2d_2[0]
[0]']
 chNormalization)

 activation_2 (Activation)   (None, 256, 256, 128)     0         ['batch_normal
ization_2[0][0]'
                                                                 ]

 conv2d_3 (Conv2D)           (None, 256, 256, 128)     147584    ['activation_2
[0][0]']

 batch_normalization_3 (Bat  (None, 256, 256, 128)     512       ['conv2d_3[0]
[0]']
 chNormalization)

 activation_3 (Activation)   (None, 256, 256, 128)     0         ['batch_normal
ization_3[0][0]'
                                                                 ]

 max_pooling2d_1 (MaxPoolin  (None, 128, 128, 128)     0         ['activation_3
[0][0]']
 g2D)

 conv2d_4 (Conv2D)           (None, 128, 128, 256)     295168    ['max_pooling2
d_1[0][0]']

 batch_normalization_4 (Bat  (None, 128, 128, 256)     1024      ['conv2d_4[0]
[0]']
```

```
 chNormalization)

 activation_4 (Activation)   (None, 128, 128, 256)        0         ['batch_normal
ization_4[0][0]'
                                                                    ]

 conv2d_5 (Conv2D)           (None, 128, 128, 256)        590080    ['activation_4
[0][0]']

 batch_normalization_5 (Bat  (None, 128, 128, 256)        1024      ['conv2d_5[0]
[0]']
 chNormalization)

 activation_5 (Activation)   (None, 128, 128, 256)        0         ['batch_normal
ization_5[0][0]'
                                                                    ]

 max_pooling2d_2 (MaxPoolin  (None, 64, 64, 256)          0         ['activation_5
[0][0]']
 g2D)

 conv2d_6 (Conv2D)           (None, 64, 64, 512)          1180160   ['max_pooling2
d_2[0][0]']

 batch_normalization_6 (Bat  (None, 64, 64, 512)          2048      ['conv2d_6[0]
[0]']
 chNormalization)

 activation_6 (Activation)   (None, 64, 64, 512)          0         ['batch_normal
ization_6[0][0]'
                                                                    ]

 conv2d_7 (Conv2D)           (None, 64, 64, 512)          2359808   ['activation_6
[0][0]']

 batch_normalization_7 (Bat  (None, 64, 64, 512)          2048      ['conv2d_7[0]
[0]']
 chNormalization)

 activation_7 (Activation)   (None, 64, 64, 512)          0         ['batch_normal
ization_7[0][0]'
                                                                    ]

 max_pooling2d_3 (MaxPoolin  (None, 32, 32, 512)          0         ['activation_7
[0][0]']
 g2D)

 conv2d_8 (Conv2D)           (None, 32, 32, 1024)         4719616   ['max_pooling2
d_3[0][0]']

 batch_normalization_8 (Bat  (None, 32, 32, 1024)         4096      ['conv2d_8[0]
[0]']
 chNormalization)

 activation_8 (Activation)   (None, 32, 32, 1024)         0         ['batch_normal
ization_8[0][0]'
                                                                    ]

 conv2d_9 (Conv2D)           (None, 32, 32, 1024)         9438208   ['activation_8
[0][0]']

 batch_normalization_9 (Bat  (None, 32, 32, 1024)         4096      ['conv2d_9[0]
[0]']
 chNormalization)
```

```
 activation_9 (Activation)   (None, 32, 32, 1024)         0         ['batch_normal
ization_9[0][0]'
                                                                    ]

 conv2d_transpose (Conv2DTr  (None, 64, 64, 512)          2097664   ['activation_9
[0][0]']
 anspose)

 concatenate (Concatenate)   (None, 64, 64, 1024)         0         ['conv2d_trans
pose[0][0]',
                                                                     'activation_7
[0][0]']

 conv2d_10 (Conv2D)          (None, 64, 64, 512)          4719104   ['concatenate
[0][0]']

 batch_normalization_10 (Ba  (None, 64, 64, 512)          2048      ['conv2d_10[0]
[0]']
 tchNormalization)

 activation_10 (Activation)  (None, 64, 64, 512)          0         ['batch_normal
ization_10[0][0]
                                                                    ']

 conv2d_11 (Conv2D)          (None, 64, 64, 512)          2359808   ['activation_1
0[0][0]']

 batch_normalization_11 (Ba  (None, 64, 64, 512)          2048      ['conv2d_11[0]
[0]']
 tchNormalization)

 activation_11 (Activation)  (None, 64, 64, 512)          0         ['batch_normal
ization_11[0][0]
                                                                    ']

 conv2d_transpose_1 (Conv2D  (None, 128, 128, 256)        524544    ['activation_1
1[0][0]']
 Transpose)

 concatenate_1 (Concatenate  (None, 128, 128, 512)        0         ['conv2d_trans
pose_1[0][0]',
 )                                                                   'activation_5
[0][0]']

 conv2d_12 (Conv2D)          (None, 128, 128, 256)        1179904   ['concatenate_
1[0][0]']

 batch_normalization_12 (Ba  (None, 128, 128, 256)        1024      ['conv2d_12[0]
[0]']
 tchNormalization)

 activation_12 (Activation)  (None, 128, 128, 256)        0         ['batch_normal
ization_12[0][0]
                                                                    ']

 conv2d_13 (Conv2D)          (None, 128, 128, 256)        590080    ['activation_1
2[0][0]']

 batch_normalization_13 (Ba  (None, 128, 128, 256)        1024      ['conv2d_13[0]
[0]']
 tchNormalization)

 activation_13 (Activation)  (None, 128, 128, 256)        0         ['batch_normal
```

```
ization_13[0][0]
                                                                  ']

 conv2d_transpose_2 (Conv2D  (None, 256, 256, 128)        131200    ['activation_1
3[0][0]']
 Transpose)

 concatenate_2 (Concatenate  (None, 256, 256, 256)        0         ['conv2d_trans
pose_2[0][0]',
 )                                                                 'activation_3
[0][0]']

 conv2d_14 (Conv2D)          (None, 256, 256, 128)        295040    ['concatenate_
2[0][0]']

 batch_normalization_14 (Ba  (None, 256, 256, 128)        512       ['conv2d_14[0]
[0]']
 tchNormalization)

 activation_14 (Activation)  (None, 256, 256, 128)        0         ['batch_normal
ization_14[0][0]
                                                                  ']

 conv2d_15 (Conv2D)          (None, 256, 256, 128)        147584    ['activation_1
4[0][0]']

 batch_normalization_15 (Ba  (None, 256, 256, 128)        512       ['conv2d_15[0]
[0]']
 tchNormalization)

 activation_15 (Activation)  (None, 256, 256, 128)        0         ['batch_normal
ization_15[0][0]
                                                                  ']

 conv2d_transpose_3 (Conv2D  (None, 512, 512, 64)         32832     ['activation_1
5[0][0]']
 Transpose)

 concatenate_3 (Concatenate  (None, 512, 512, 128)        0         ['conv2d_trans
pose_3[0][0]',
 )                                                                 'activation_1
[0][0]']

 conv2d_16 (Conv2D)          (None, 512, 512, 64)         73792     ['concatenate_
3[0][0]']

 batch_normalization_16 (Ba  (None, 512, 512, 64)         256       ['conv2d_16[0]
[0]']
 tchNormalization)

 activation_16 (Activation)  (None, 512, 512, 64)         0         ['batch_normal
ization_16[0][0]
                                                                  ']

 conv2d_17 (Conv2D)          (None, 512, 512, 64)         36928     ['activation_1
6[0][0]']

 batch_normalization_17 (Ba  (None, 512, 512, 64)         256       ['conv2d_17[0]
[0]']
 tchNormalization)

 activation_17 (Activation)  (None, 512, 512, 64)         0         ['batch_normal
ization_17[0][0]
                                                                  ']
```

```
 conv2d_18 (Conv2D)          (None, 512, 512, 1)          65        ['activation_1
7[0][0]']

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

In [17]:
```
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 [==============================] - 69s 3s/step - loss: 1.4584 - accuracy: 0.53
81 - val_loss: -59.4691 - val_accuracy: 0.0390
Epoch 2/100
6/6 [==============================] - 6s 1s/step - loss: -4.0780 - accuracy: 0.55
93 - val_loss: -52.5994 - val_accuracy: 0.0406
Epoch 3/100
6/6 [==============================] - 6s 1s/step - loss: -9.8813 - accuracy: 0.57
44 - val_loss: -51.5387 - val_accuracy: 0.0387
Epoch 4/100
6/6 [==============================] - 6s 1s/step - loss: -14.8992 - accuracy: 0.5
871 - val_loss: -52.7015 - val_accuracy: 0.0374
Epoch 5/100
6/6 [==============================] - 6s 1s/step - loss: -20.8392 - accuracy: 0.5
960 - val_loss: -54.5648 - val_accuracy: 0.0390
Epoch 6/100
6/6 [==============================] - 6s 1s/step - loss: -25.5299 - accuracy: 0.6
010 - val_loss: -56.6846 - val_accuracy: 0.0441
Epoch 7/100
6/6 [==============================] - 6s 1s/step - loss: -32.6551 - accuracy: 0.6
067 - val_loss: -59.4712 - val_accuracy: 0.0496
Epoch 8/100
6/6 [==============================] - 6s 1s/step - loss: -38.2750 - accuracy: 0.6
096 - val_loss: -61.7416 - val_accuracy: 0.0589
Epoch 9/100
6/6 [==============================] - 6s 1s/step - loss: -42.4987 - accuracy: 0.6
135 - val_loss: -64.5789 - val_accuracy: 0.0713
Epoch 10/100
6/6 [==============================] - 6s 1s/step - loss: -49.6998 - accuracy: 0.6
146 - val_loss: -66.7110 - val_accuracy: 0.0890
Epoch 11/100
6/6 [==============================] - 6s 1s/step - loss: -56.2138 - accuracy: 0.6
179 - val_loss: -68.6948 - val_accuracy: 0.1110
Epoch 12/100
6/6 [==============================] - 6s 1s/step - loss: -61.8829 - accuracy: 0.6
198 - val_loss: -70.0121 - val_accuracy: 0.1378
Epoch 13/100
6/6 [==============================] - 7s 1s/step - loss: -68.1216 - accuracy: 0.6
215 - val_loss: -70.0266 - val_accuracy: 0.1709
Epoch 14/100
6/6 [==============================] - 6s 1s/step - loss: -75.1829 - accuracy: 0.6
221 - val_loss: -69.7694 - val_accuracy: 0.2076
Epoch 15/100
6/6 [==============================] - 7s 1s/step - loss: -81.4861 - accuracy: 0.6
246 - val_loss: -69.4329 - val_accuracy: 0.2449
Epoch 16/100
6/6 [==============================] - 7s 1s/step - loss: -87.2625 - accuracy: 0.6
254 - val_loss: -68.7621 - val_accuracy: 0.2838
Epoch 17/100
6/6 [==============================] - 7s 1s/step - loss: -95.6262 - accuracy: 0.6
329 - val_loss: -69.2458 - val_accuracy: 0.3190
Epoch 18/100
6/6 [==============================] - 7s 1s/step - loss: -106.2518 - accuracy: 0.
6378 - val_loss: -70.3302 - val_accuracy: 0.3492
Epoch 19/100
6/6 [==============================] - 7s 1s/step - loss: -109.1521 - accuracy: 0.
6406 - val_loss: -71.2003 - val_accuracy: 0.3781
Epoch 20/100
6/6 [==============================] - 7s 1s/step - loss: -118.9458 - accuracy: 0.
6454 - val_loss: -70.6142 - val_accuracy: 0.4099
Epoch 21/100
6/6 [==============================] - 7s 1s/step - loss: -123.0312 - accuracy: 0.
6460 - val_loss: -69.7355 - val_accuracy: 0.4390
Epoch 22/100
```

```
6/6 [==============================] - 7s 1s/step - loss: -131.2928 - accuracy: 0.
6534 - val_loss: -69.1865 - val_accuracy: 0.4655
Epoch 23/100
6/6 [==============================] - 7s 1s/step - loss: -135.1480 - accuracy: 0.
6542 - val_loss: -68.8326 - val_accuracy: 0.4895
Epoch 24/100
6/6 [==============================] - 6s 1s/step - loss: -142.9601 - accuracy: 0.
6575 - val_loss: -69.4171 - val_accuracy: 0.5099
Epoch 25/100
6/6 [==============================] - 7s 1s/step - loss: -151.2700 - accuracy: 0.
6563 - val_loss: -69.2733 - val_accuracy: 0.5308
Epoch 26/100
6/6 [==============================] - 7s 1s/step - loss: -155.3525 - accuracy: 0.
6634 - val_loss: -68.6408 - val_accuracy: 0.5505
Epoch 27/100
6/6 [==============================] - 7s 1s/step - loss: -155.2447 - accuracy: 0.
6603 - val_loss: -69.4871 - val_accuracy: 0.5667
Epoch 28/100
6/6 [==============================] - 7s 1s/step - loss: -168.5661 - accuracy: 0.
6687 - val_loss: -68.8839 - val_accuracy: 0.5854
Epoch 29/100
6/6 [==============================] - 7s 1s/step - loss: -171.1983 - accuracy: 0.
6693 - val_loss: -67.0131 - val_accuracy: 0.6041
Epoch 30/100
6/6 [==============================] - 7s 1s/step - loss: -176.8731 - accuracy: 0.
6705 - val_loss: -67.6317 - val_accuracy: 0.6173
Epoch 31/100
6/6 [==============================] - 7s 1s/step - loss: -175.9806 - accuracy: 0.
6719 - val_loss: -68.7374 - val_accuracy: 0.6288
Epoch 32/100
6/6 [==============================] - 7s 1s/step - loss: -184.6329 - accuracy: 0.
6747 - val_loss: -68.5022 - val_accuracy: 0.6416
Epoch 33/100
6/6 [==============================] - 7s 1s/step - loss: -187.6351 - accuracy: 0.
6784 - val_loss: -68.4204 - val_accuracy: 0.6535
Epoch 34/100
6/6 [==============================] - 7s 1s/step - loss: -195.0103 - accuracy: 0.
6810 - val_loss: -69.9387 - val_accuracy: 0.6626
Epoch 35/100
6/6 [==============================] - 7s 1s/step - loss: -192.1151 - accuracy: 0.
6802 - val_loss: -67.3500 - val_accuracy: 0.6756
Epoch 36/100
6/6 [==============================] - 7s 1s/step - loss: -198.6129 - accuracy: 0.
6811 - val_loss: -68.0950 - val_accuracy: 0.6839
Epoch 37/100
6/6 [==============================] - 7s 1s/step - loss: -205.5051 - accuracy: 0.
6839 - val_loss: -72.1549 - val_accuracy: 0.6870
Epoch 38/100
6/6 [==============================] - 6s 1s/step - loss: -209.4695 - accuracy: 0.
6886 - val_loss: -73.7014 - val_accuracy: 0.6928
Epoch 39/100
6/6 [==============================] - 7s 1s/step - loss: -212.7164 - accuracy: 0.
6887 - val_loss: -75.5219 - val_accuracy: 0.6972
Epoch 40/100
6/6 [==============================] - 6s 1s/step - loss: -212.9200 - accuracy: 0.
6875 - val_loss: -70.4235 - val_accuracy: 0.7094
Epoch 41/100
6/6 [==============================] - 6s 1s/step - loss: -210.4420 - accuracy: 0.
6922 - val_loss: -72.5453 - val_accuracy: 0.7130
Epoch 42/100
6/6 [==============================] - 7s 1s/step - loss: -221.3136 - accuracy: 0.
6960 - val_loss: -80.3664 - val_accuracy: 0.7106
Epoch 43/100
6/6 [==============================] - 6s 1s/step - loss: -225.6913 - accuracy: 0.
```

```
6931 - val_loss: -84.7790 - val_accuracy: 0.7125
Epoch 44/100
6/6 [==============================] - 7s 1s/step - loss: -226.7475 - accuracy: 0.
6976 - val_loss: -75.1528 - val_accuracy: 0.7252
Epoch 45/100
6/6 [==============================] - 7s 1s/step - loss: -232.6748 - accuracy: 0.
7002 - val_loss: -85.0264 - val_accuracy: 0.7207
Epoch 46/100
6/6 [==============================] - 7s 1s/step - loss: -234.5686 - accuracy: 0.
7030 - val_loss: -93.3619 - val_accuracy: 0.7175
Epoch 47/100
6/6 [==============================] - 7s 1s/step - loss: -227.7934 - accuracy: 0.
7007 - val_loss: -82.5681 - val_accuracy: 0.7325
Epoch 48/100
6/6 [==============================] - 7s 1s/step - loss: -235.7104 - accuracy: 0.
7033 - val_loss: -84.8233 - val_accuracy: 0.7329
Epoch 49/100
6/6 [==============================] - 7s 1s/step - loss: -242.1422 - accuracy: 0.
7077 - val_loss: -107.8995 - val_accuracy: 0.7200
Epoch 50/100
6/6 [==============================] - 7s 1s/step - loss: -238.6613 - accuracy: 0.
7064 - val_loss: -99.7993 - val_accuracy: 0.7295
Epoch 51/100
6/6 [==============================] - 7s 1s/step - loss: -246.6499 - accuracy: 0.
7072 - val_loss: -90.9447 - val_accuracy: 0.7389
Epoch 52/100
6/6 [==============================] - 7s 1s/step - loss: -251.4989 - accuracy: 0.
7119 - val_loss: -106.7262 - val_accuracy: 0.7299
Epoch 53/100
6/6 [==============================] - 7s 1s/step - loss: -250.1782 - accuracy: 0.
7119 - val_loss: -94.1565 - val_accuracy: 0.7425
Epoch 54/100
6/6 [==============================] - 7s 1s/step - loss: -254.8246 - accuracy: 0.
7149 - val_loss: -106.2364 - val_accuracy: 0.7363
Epoch 55/100
6/6 [==============================] - 7s 1s/step - loss: -255.8452 - accuracy: 0.
7145 - val_loss: -109.3358 - val_accuracy: 0.7351
Epoch 56/100
6/6 [==============================] - 7s 1s/step - loss: -256.2433 - accuracy: 0.
7151 - val_loss: -125.4623 - val_accuracy: 0.7278
Epoch 57/100
6/6 [==============================] - 7s 1s/step - loss: -264.4292 - accuracy: 0.
7194 - val_loss: -119.9042 - val_accuracy: 0.7337
Epoch 58/100
6/6 [==============================] - 7s 1s/step - loss: -264.7392 - accuracy: 0.
7182 - val_loss: -120.4300 - val_accuracy: 0.7347
Epoch 59/100
6/6 [==============================] - 7s 1s/step - loss: -260.6508 - accuracy: 0.
7173 - val_loss: -135.3573 - val_accuracy: 0.7282
Epoch 60/100
6/6 [==============================] - 7s 1s/step - loss: -265.1877 - accuracy: 0.
7202 - val_loss: -120.0751 - val_accuracy: 0.7401
Epoch 61/100
6/6 [==============================] - 7s 1s/step - loss: -267.3077 - accuracy: 0.
7207 - val_loss: -133.3182 - val_accuracy: 0.7338
Epoch 62/100
6/6 [==============================] - 7s 1s/step - loss: -272.4491 - accuracy: 0.
7251 - val_loss: -148.0474 - val_accuracy: 0.7277
Epoch 63/100
6/6 [==============================] - 7s 1s/step - loss: -269.0103 - accuracy: 0.
7230 - val_loss: -155.7419 - val_accuracy: 0.7218
Epoch 64/100
6/6 [==============================] - 7s 1s/step - loss: -273.5331 - accuracy: 0.
7232 - val_loss: -118.0016 - val_accuracy: 0.7481
```

```
Epoch 65/100
6/6 [==============================] - 7s 1s/step - loss: -273.5718 - accuracy: 0.
7256 - val_loss: -149.9573 - val_accuracy: 0.7303
Epoch 66/100
6/6 [==============================] - 7s 1s/step - loss: -276.2815 - accuracy: 0.
7302 - val_loss: -175.2160 - val_accuracy: 0.7157
Epoch 67/100
6/6 [==============================] - 7s 1s/step - loss: -279.7736 - accuracy: 0.
7296 - val_loss: -130.6171 - val_accuracy: 0.7448
Epoch 68/100
6/6 [==============================] - 7s 1s/step - loss: -276.4866 - accuracy: 0.
7277 - val_loss: -167.1375 - val_accuracy: 0.7204
Epoch 69/100
6/6 [==============================] - 7s 1s/step - loss: -278.3327 - accuracy: 0.
7279 - val_loss: -178.9592 - val_accuracy: 0.7196
Epoch 70/100
6/6 [==============================] - 7s 1s/step - loss: -283.1725 - accuracy: 0.
7323 - val_loss: -178.3538 - val_accuracy: 0.7160
Epoch 71/100
6/6 [==============================] - 7s 1s/step - loss: -280.7878 - accuracy: 0.
7294 - val_loss: -171.5546 - val_accuracy: 0.7264
Epoch 72/100
6/6 [==============================] - 6s 1s/step - loss: -286.0898 - accuracy: 0.
7330 - val_loss: -181.2969 - val_accuracy: 0.7188
Epoch 73/100
6/6 [==============================] - 7s 1s/step - loss: -277.0054 - accuracy: 0.
7313 - val_loss: -142.9664 - val_accuracy: 0.7472
Epoch 74/100
6/6 [==============================] - 7s 1s/step - loss: -285.3331 - accuracy: 0.
7343 - val_loss: -211.9692 - val_accuracy: 0.7003
Epoch 75/100
6/6 [==============================] - 7s 1s/step - loss: -289.3420 - accuracy: 0.
7351 - val_loss: -166.3372 - val_accuracy: 0.7349
Epoch 76/100
6/6 [==============================] - 7s 1s/step - loss: -287.9325 - accuracy: 0.
7357 - val_loss: -190.6336 - val_accuracy: 0.7171
Epoch 77/100
6/6 [==============================] - 7s 1s/step - loss: -289.8391 - accuracy: 0.
7361 - val_loss: -184.5650 - val_accuracy: 0.7250
Epoch 78/100
6/6 [==============================] - 7s 1s/step - loss: -291.3858 - accuracy: 0.
7381 - val_loss: -179.5513 - val_accuracy: 0.7257
Epoch 79/100
6/6 [==============================] - 6s 1s/step - loss: -292.4685 - accuracy: 0.
7381 - val_loss: -188.6294 - val_accuracy: 0.7224
Epoch 80/100
6/6 [==============================] - 7s 1s/step - loss: -296.1159 - accuracy: 0.
7375 - val_loss: -214.9561 - val_accuracy: 0.7073
Epoch 81/100
6/6 [==============================] - 7s 1s/step - loss: -292.1382 - accuracy: 0.
7374 - val_loss: -173.8280 - val_accuracy: 0.7312
Epoch 82/100
6/6 [==============================] - 7s 1s/step - loss: -295.0531 - accuracy: 0.
7389 - val_loss: -214.0993 - val_accuracy: 0.7092
Epoch 83/100
6/6 [==============================] - 7s 1s/step - loss: -299.5427 - accuracy: 0.
7425 - val_loss: -189.0947 - val_accuracy: 0.7243
Epoch 84/100
6/6 [==============================] - 7s 1s/step - loss: -299.1367 - accuracy: 0.
7418 - val_loss: -218.3246 - val_accuracy: 0.7101
Epoch 85/100
6/6 [==============================] - 7s 1s/step - loss: -298.7264 - accuracy: 0.
7421 - val_loss: -199.0284 - val_accuracy: 0.7209
Epoch 86/100
```

```
6/6 [==============================] - 7s 1s/step - loss: -296.4487 - accuracy: 0.
7408 - val_loss: -199.5837 - val_accuracy: 0.7219
Epoch 87/100
6/6 [==============================] - 7s 1s/step - loss: -297.0472 - accuracy: 0.
7416 - val_loss: -201.7486 - val_accuracy: 0.7188
Epoch 88/100
6/6 [==============================] - 7s 1s/step - loss: -305.0974 - accuracy: 0.
7446 - val_loss: -183.2762 - val_accuracy: 0.7324
Epoch 89/100
6/6 [==============================] - 7s 1s/step - loss: -302.0031 - accuracy: 0.
7436 - val_loss: -217.2551 - val_accuracy: 0.7130
Epoch 90/100
6/6 [==============================] - 7s 1s/step - loss: -304.9904 - accuracy: 0.
7465 - val_loss: -199.2539 - val_accuracy: 0.7232
Epoch 91/100
6/6 [==============================] - 7s 1s/step - loss: -299.6154 - accuracy: 0.
7448 - val_loss: -204.5028 - val_accuracy: 0.7208
Epoch 92/100
6/6 [==============================] - 7s 1s/step - loss: -307.3618 - accuracy: 0.
7480 - val_loss: -209.1508 - val_accuracy: 0.7186
Epoch 93/100
6/6 [==============================] - 7s 1s/step - loss: -306.7337 - accuracy: 0.
7464 - val_loss: -206.7167 - val_accuracy: 0.7232
Epoch 94/100
6/6 [==============================] - 6s 1s/step - loss: -303.5547 - accuracy: 0.
7461 - val_loss: -197.9530 - val_accuracy: 0.7269
Epoch 95/100
6/6 [==============================] - 7s 1s/step - loss: -306.8899 - accuracy: 0.
7471 - val_loss: -205.5994 - val_accuracy: 0.7253
Epoch 96/100
6/6 [==============================] - 7s 1s/step - loss: -312.5930 - accuracy: 0.
7491 - val_loss: -227.2711 - val_accuracy: 0.7115
Epoch 97/100
6/6 [==============================] - 7s 1s/step - loss: -313.0431 - accuracy: 0.
7490 - val_loss: -201.3752 - val_accuracy: 0.7286
Epoch 98/100
6/6 [==============================] - 7s 1s/step - loss: -313.7921 - accuracy: 0.
7507 - val_loss: -205.3426 - val_accuracy: 0.7225
Epoch 99/100
6/6 [==============================] - 7s 1s/step - loss: -311.1559 - accuracy: 0.
7471 - val_loss: -235.0909 - val_accuracy: 0.7089
Epoch 100/100
6/6 [==============================] - 7s 1s/step - loss: -315.2231 - accuracy: 0.
7510 - val_loss: -216.9062 - val_accuracy: 0.7166
```

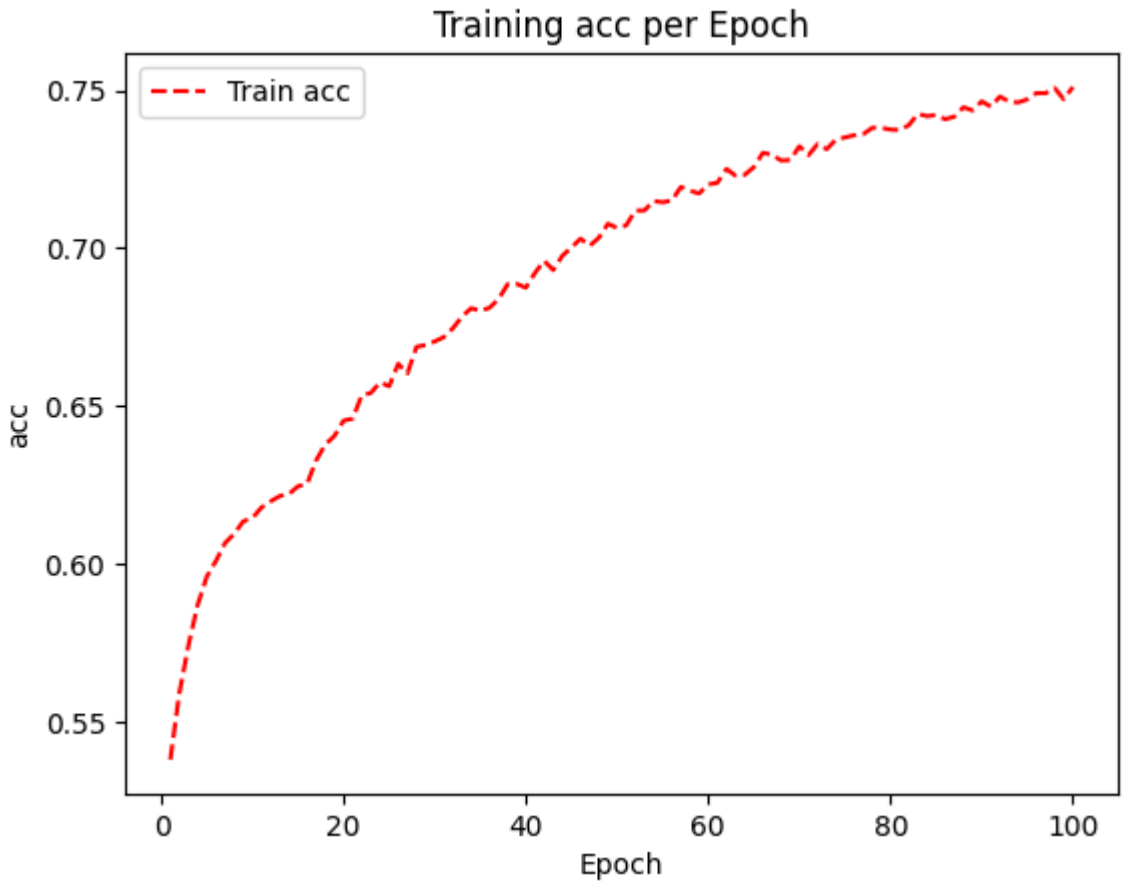
In [19]:

```
print(unet_result.history.keys())
import matplotlib.pyplot as plt
f, ax = plt.subplots()
ax.plot([None] + unet_result.history['accuracy'], 'r--')
# Plot legend and use the best location automatically: loc = 0.
ax.legend(['Train acc'], loc = 0)
ax.set_title('Training acc per Epoch')
ax.set_xlabel('Epoch')
ax.set_ylabel('acc')
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

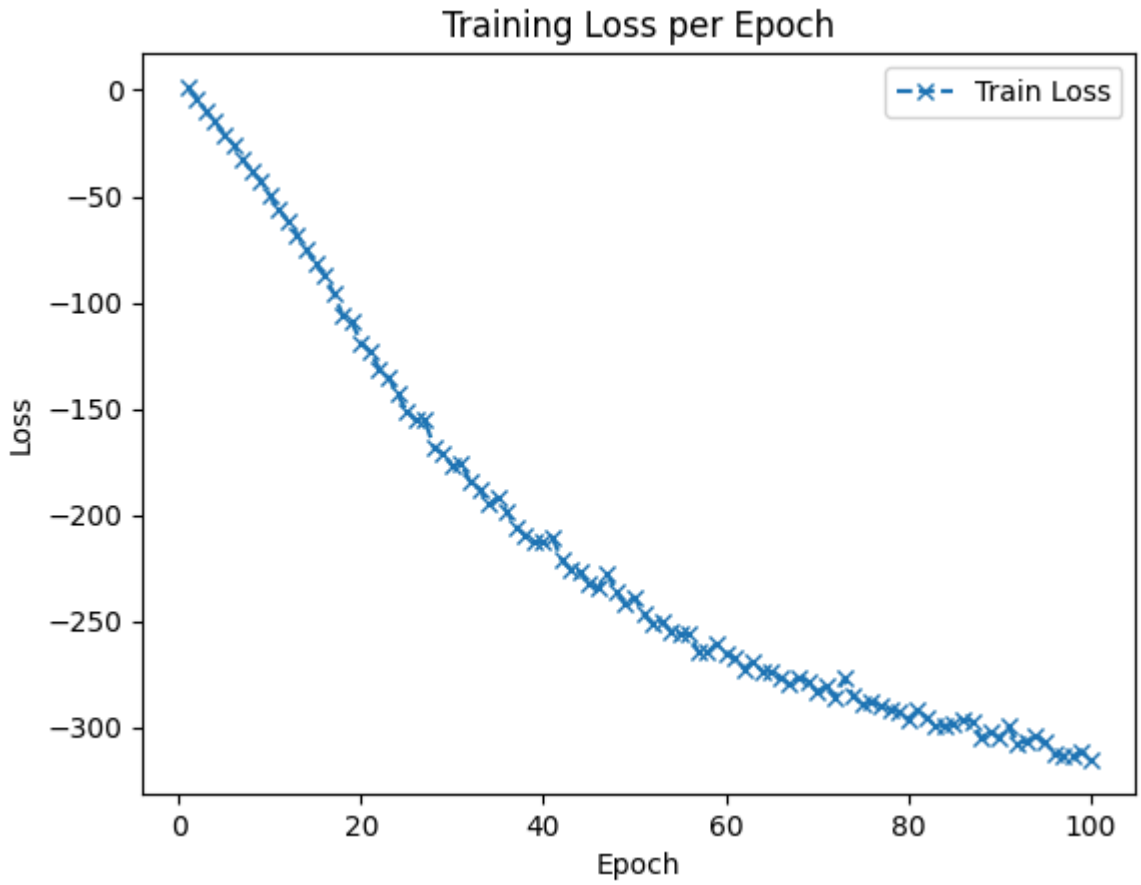
Out[19]:

```
Text(0, 0.5, 'acc')
```



In [20]:
```
f, ax = plt.subplots()
ax.plot([None] + unet_result.history['loss'], 'x--')
# Plot legend and use the best location automatically: loc = 0.
ax.legend(['Train Loss'], loc = 0)
ax.set_title('Training Loss per Epoch')
ax.set_xlabel('Epoch')
ax.set_ylabel('Loss')
```

Out[20]:
```
Text(0, 0.5, 'Loss')
```



In [21]:
```
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 [22]:
```
unet_predict = unet_model.predict(images_test)
```

```
1/1 [==============================] - 19s 19s/step
```

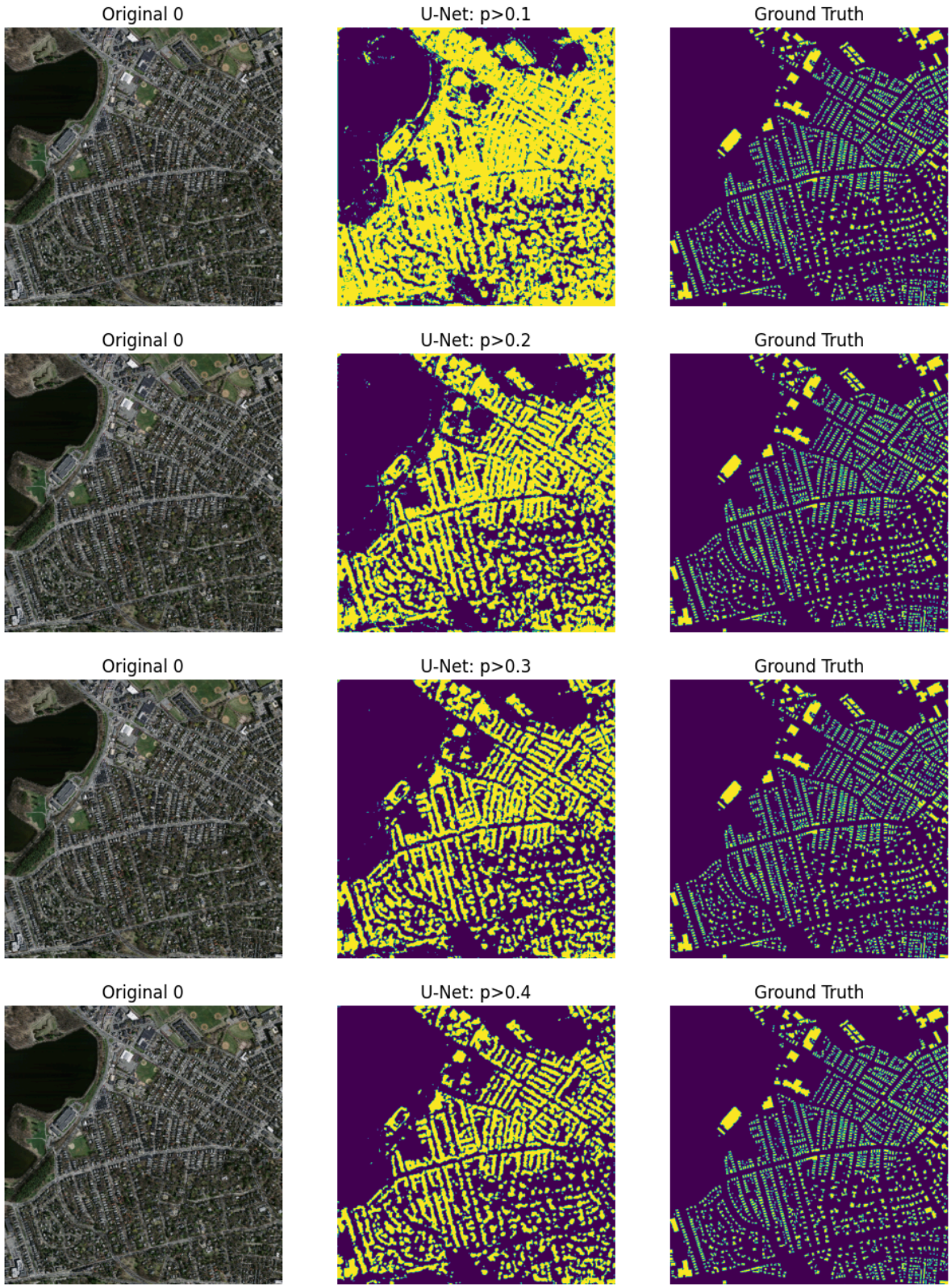
In [23]:
```
len(images_test)
```

Out[23]:
```
6
```

In [24]:
```
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)
```

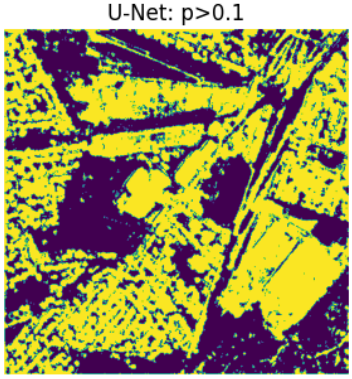
In [25]:
```
show_test_idx = random.sample(range(len(unet_predict)), 3)
for idx in show_test_idx:
    show_result(idx, images_test[idx], unet_predict1[idx], masks_test[idx], 0.1)
    show_result(idx, images_test[idx], unet_predict2[idx], masks_test[idx], 0.2)
```

```
    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()
```

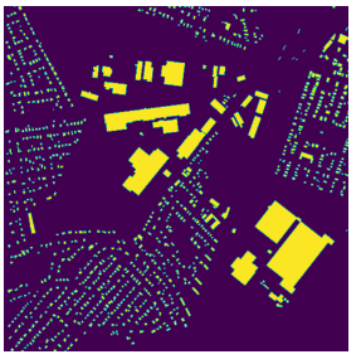


Original 3

U-Net: p>0.1

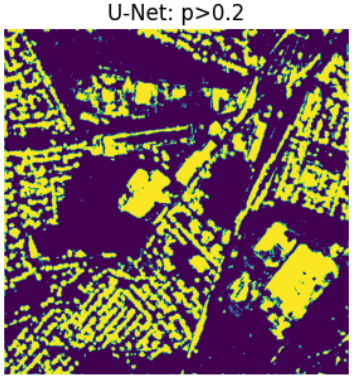


Ground Truth

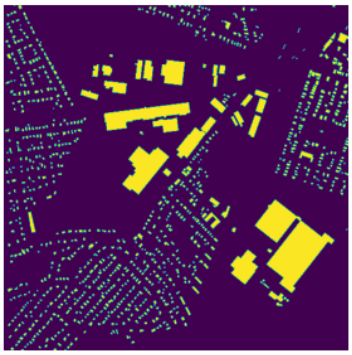


Original 3

U-Net: p>0.2

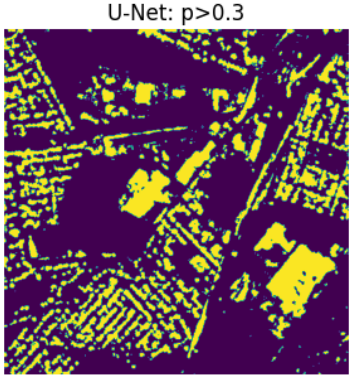


Ground Truth

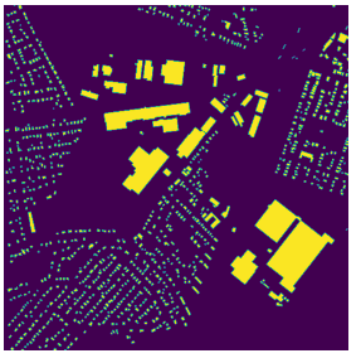


Original 3

U-Net: p>0.3

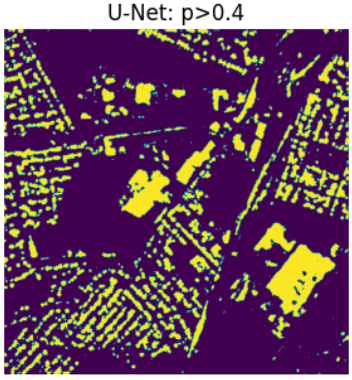


Ground Truth

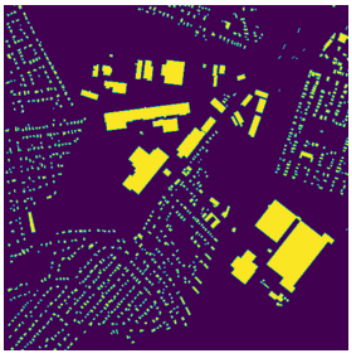


Original 3

U-Net: p>0.4

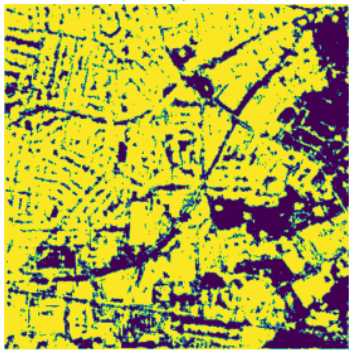


Ground Truth

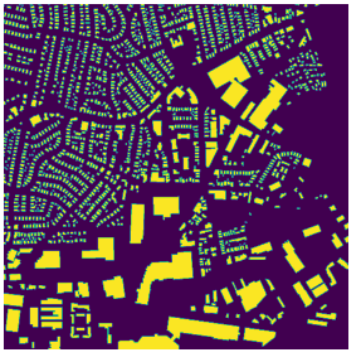


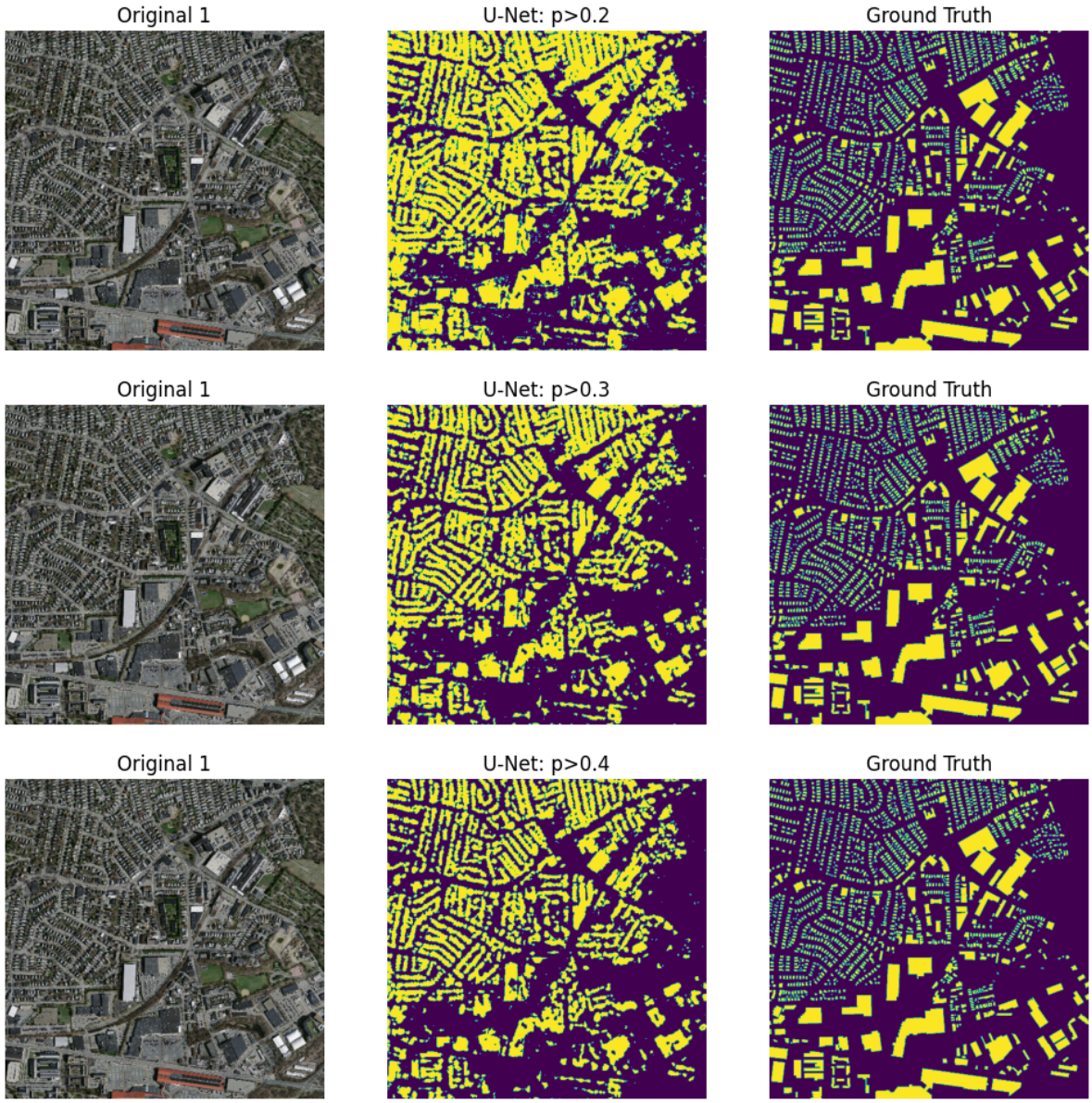
Original 1

U-Net: p>0.1



Ground Truth





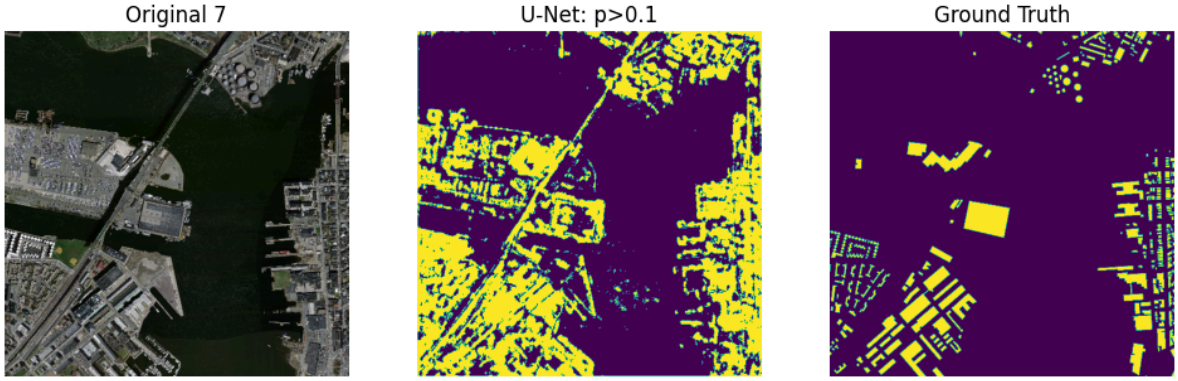
In [26]:
```
unet_predictT = unet_model.predict(test_images)
```

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

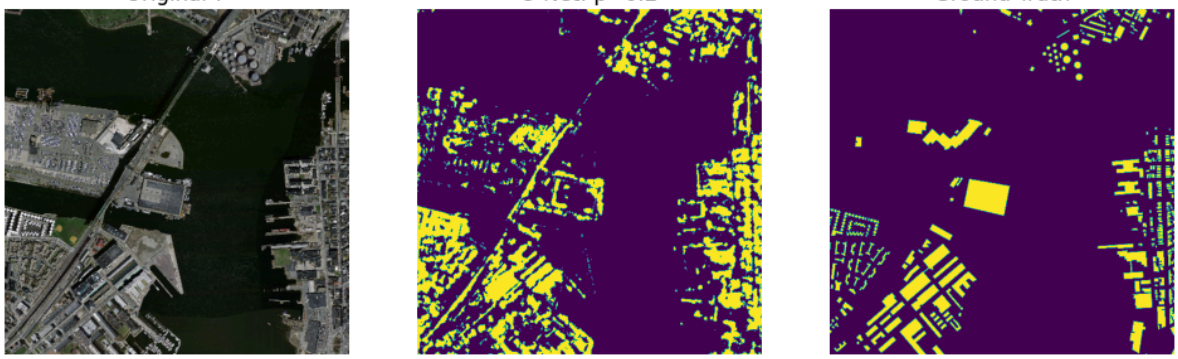
In [27]:
```
unet_predict1 = (unet_predictT > 0.1).astype(np.uint8)
unet_predict2 = (unet_predictT > 0.2).astype(np.uint8)
unet_predict3 = (unet_predictT > 0.3).astype(np.uint8)
unet_predict4 = (unet_predictT > 0.4).astype(np.uint8)
```

In [28]:
```
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()
```

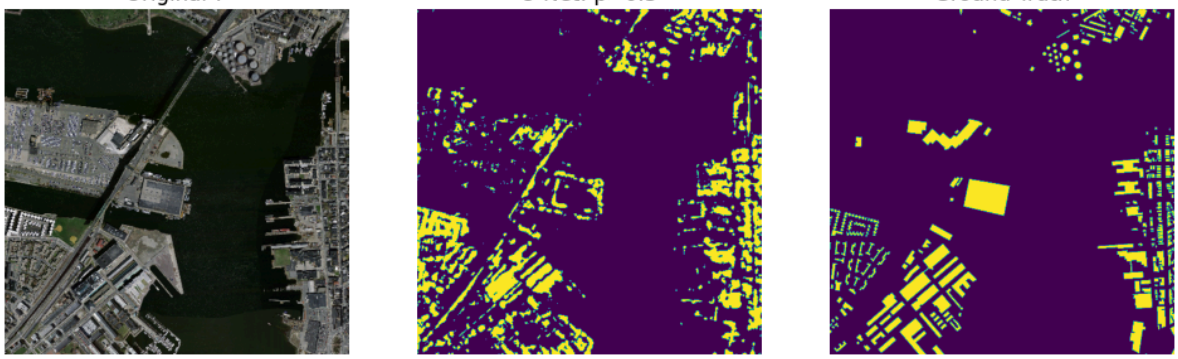
Original 7 U-Net: p>0.1 Ground Truth



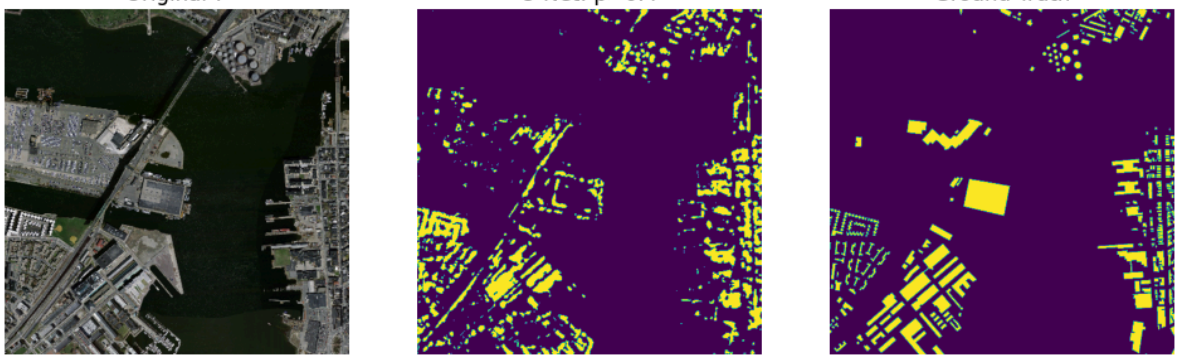
Original 7 U-Net: p>0.2 Ground Truth



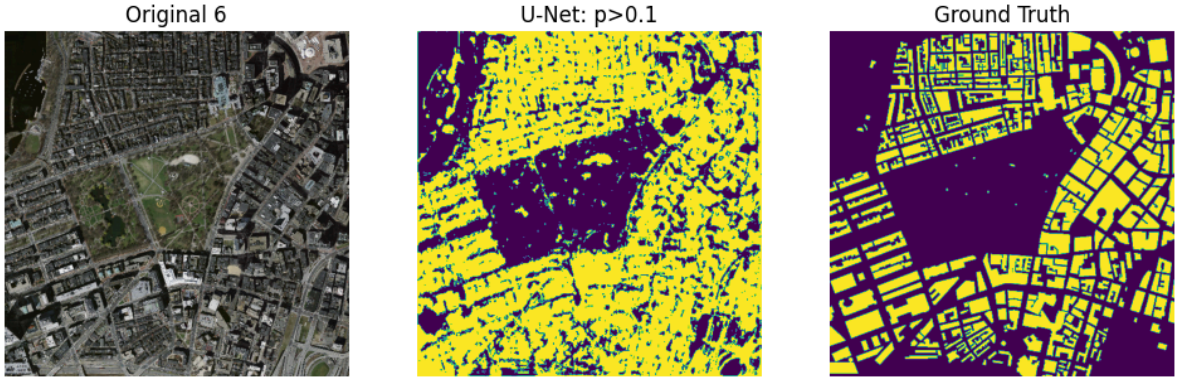
Original 7 U-Net: p>0.3 Ground Truth



Original 7 U-Net: p>0.4 Ground Truth

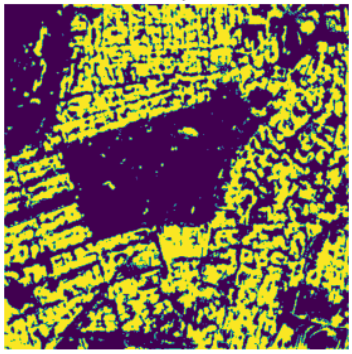


Original 6 U-Net: p>0.1 Ground Truth

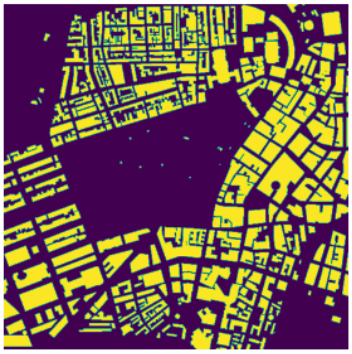


Original 6

U-Net: p>0.2

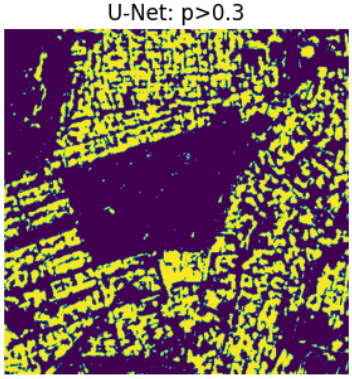


Ground Truth

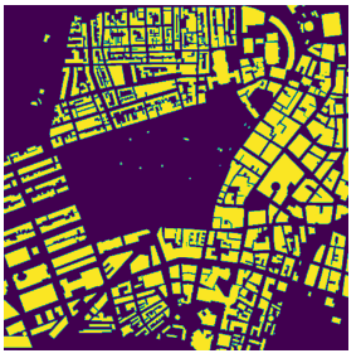


Original 6

U-Net: p>0.3

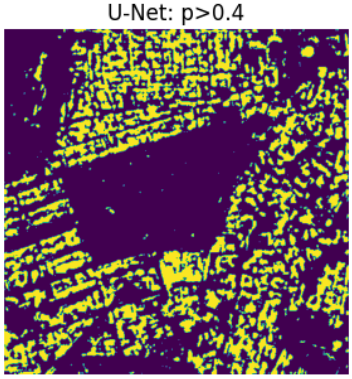


Ground Truth

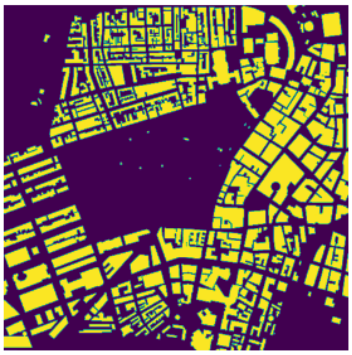


Original 6

U-Net: p>0.4



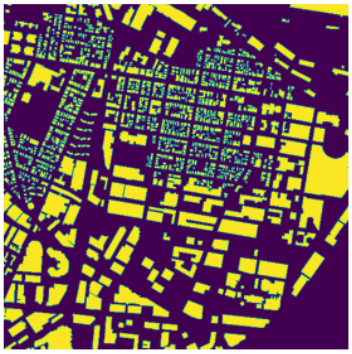
Ground Truth



Original 3

U-Net: p>0.1

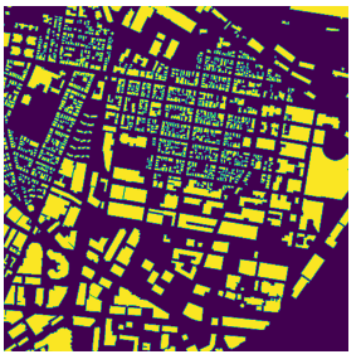
Ground Truth

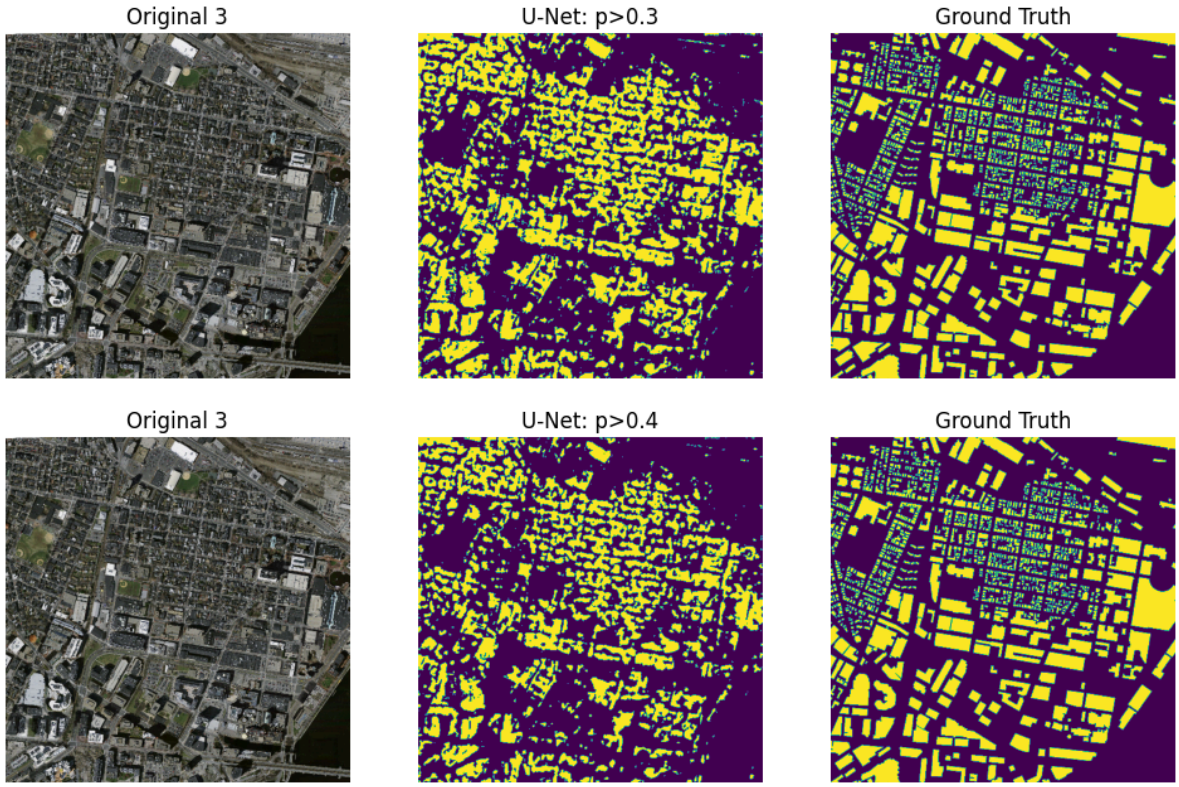


Original 3

U-Net: p>0.2

Ground Truth




Original 3
U-Net: p>0.3
Ground Truth
Original 3
U-Net: p>0.4
Ground Truth

In [ ]: