월리를 찾아라

Find Wally with CNN

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import keras.layers as layers
import keras.optimizers as optimizers
from keras.models import Model, load_model
from keras.callbacks import LambdaCallback, ModelCheckpoint, ReduceLROnPlateau
import seaborn as sns
from PIL import Image
from skimage.transform import resize

import threading, random, os
from keras.utils.np_utils import to_categorical
from tensorflow.keras import optimizers
```

1 . matplotlib : 그래프 패키지

2 . patches : 사각형 그려줌

3 . seaborn : 그래프 패키지

4 . PIL(Pillow) : 이미지 처리 패키지

5 . skimage : 이미지 처리 패키지

Load DataSet 이미지 data를 가져옴

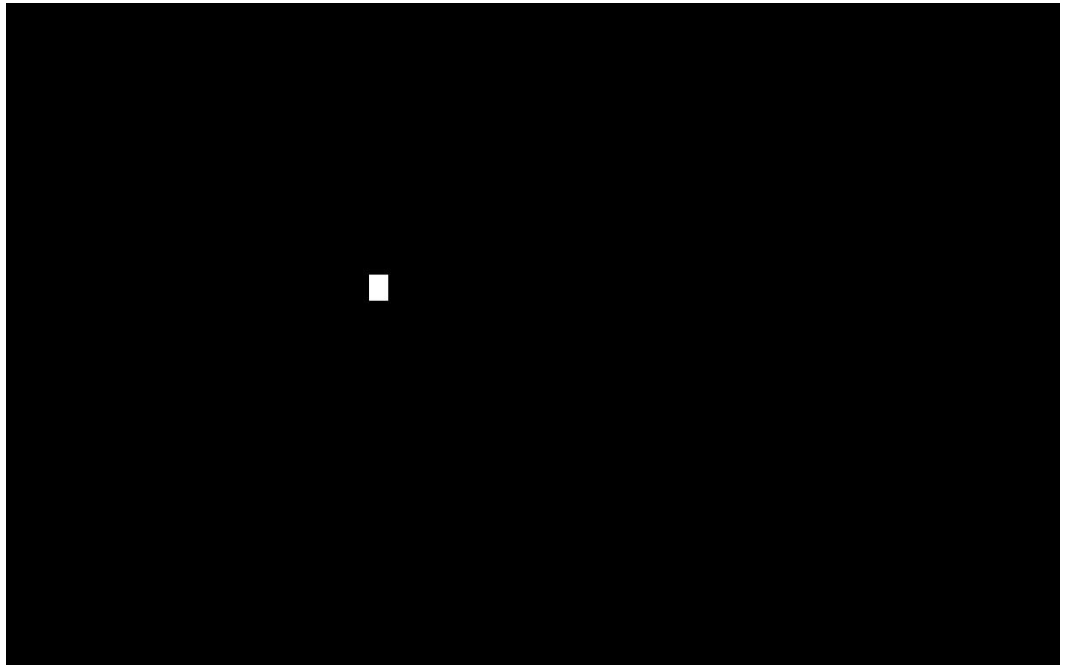
Imgs_unit8.npy : 전체 원본 이미지

labels_unit8.npy : 전체 원본 이미지 마스크(월리가 있는 곳만 255, 나머지 0)

waldo_sub_imgs_unit8.npy : 월리 이미지

Waldo_sub_labels_unit8.npy : 월리 이미지 마스크

전체 원본 이미지



전체 원본 이미지 마스크

Load DataSet 이미지 data를 가져옴

Imgs_unit8.npy : 전체 원본 이미지
labels_unit8.npy : 전체 원본 이미지 마스크(월리가 있는 곳만 255, 나머지 0)
waldo_sub_imgs_unit8.npy : 월리 이미지
Waldo_sub_labels_unit8.npy : 월리 이미지 마스크

월리 이미지

월리 이미지 마스크

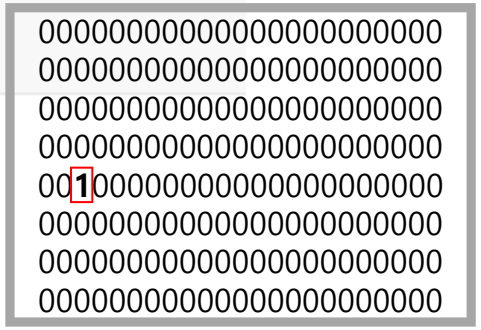
Load DataSet 이미지 data를 가져옴

```
imgs = np.load('dataset/imgs_uint8.npy', allow_pickle=True).astype(np.float32) / 255.
labels = np.load('dataset/labels_uint8.npy', allow_pickle=True).astype(np.float32) / 255.
waldo_sub_imgs = np.load('dataset/waldo_sub_imgs_uint8.npy', allow_pickle=True) / 255.
waldo_sub_labels = np.load('dataset/waldo_sub_labels_uint8.npy', allow_pickle=True) / 255.

print(imgs.shape, labels.shape)
print(waldo_sub_imgs.shape, waldo_sub_labels.shape)
```

18개 이미지, 가로 2800, 세로 1760의 컬러 이미지

```
(18, 1760, 2800, 3) (18, 1760, 2800)
(18,) (18,)
```

라벨 : 원본 이미지와 같은 사이즈 이미지로
월리가 있는 곳은 1, 없는 곳은 0으로 표시



Data Generator 이미지를 무작위 자르고, 뒤집는다.

```
def get_slice(self, i,o):
    start = random.randint(0, i-o) if self.train else (i-o)
    return slice(start, start+o)

def get_item(self, idx):
    slice_r = self.get_slice(self.ri[idx], self.ro)
    slice_c = self.get_slice(self.ci[idx], self.co)
    x = self.x[idx][slice_r, slice_c]
    y = self.y[idx][slice_r, slice_c]
    if self.train and (random.random()>0.5):
        y = y[:,::-1]
        x = x[:,::-1]
    if not self.waldo and np.sum(y)==0:
        return None

    return x, to_categorical(y, num_classes=2).reshape((y.shape[0] * y.shape[1], 2))
```

랜덤으로 이미지를 자른다.

랜덤으로 좌우로 뒤집는다.

0.5의 확률로 좌우를 반전

Data Generator 이미지를 무작위 자르고, 뒤집는다.

224

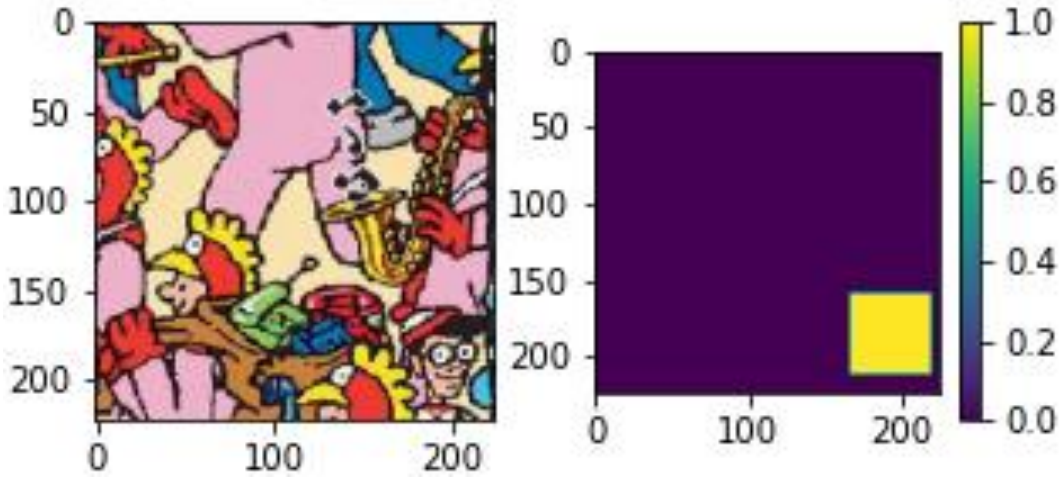
224

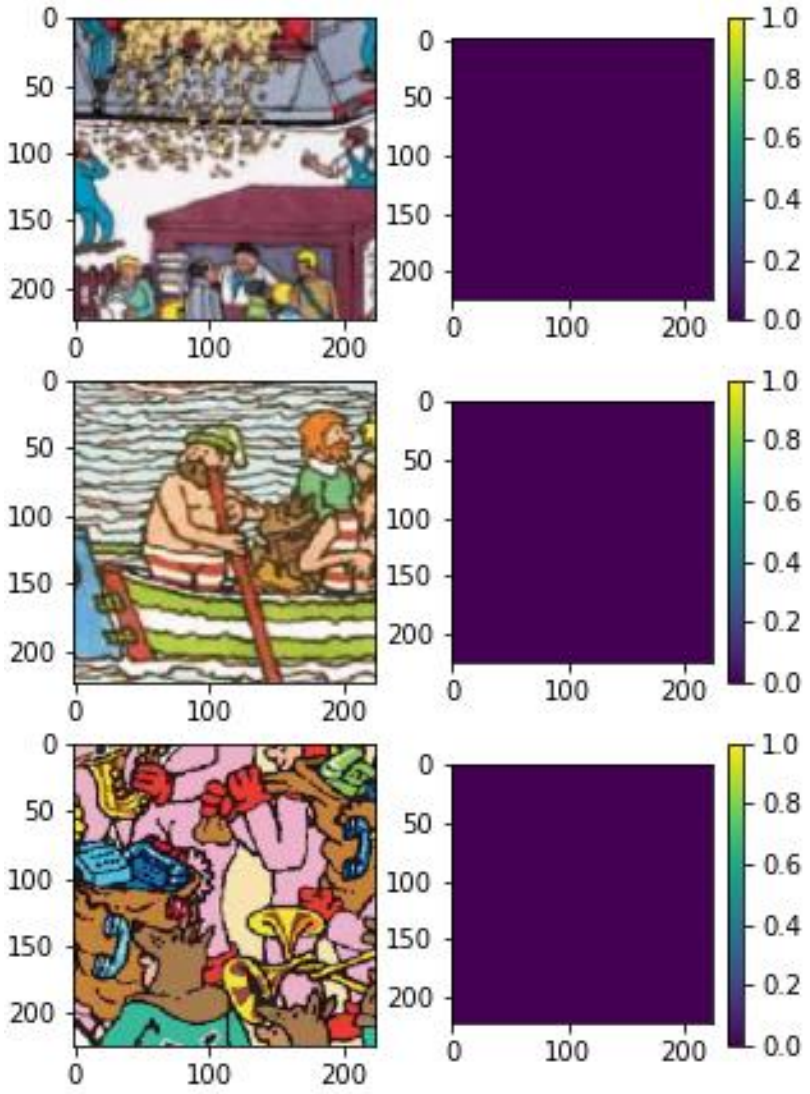
Preview Sample Pannel Images 이미지 샘플 확인

1/3은 월리가 있는 이미지, 나머지는 월리가 없는 이미지

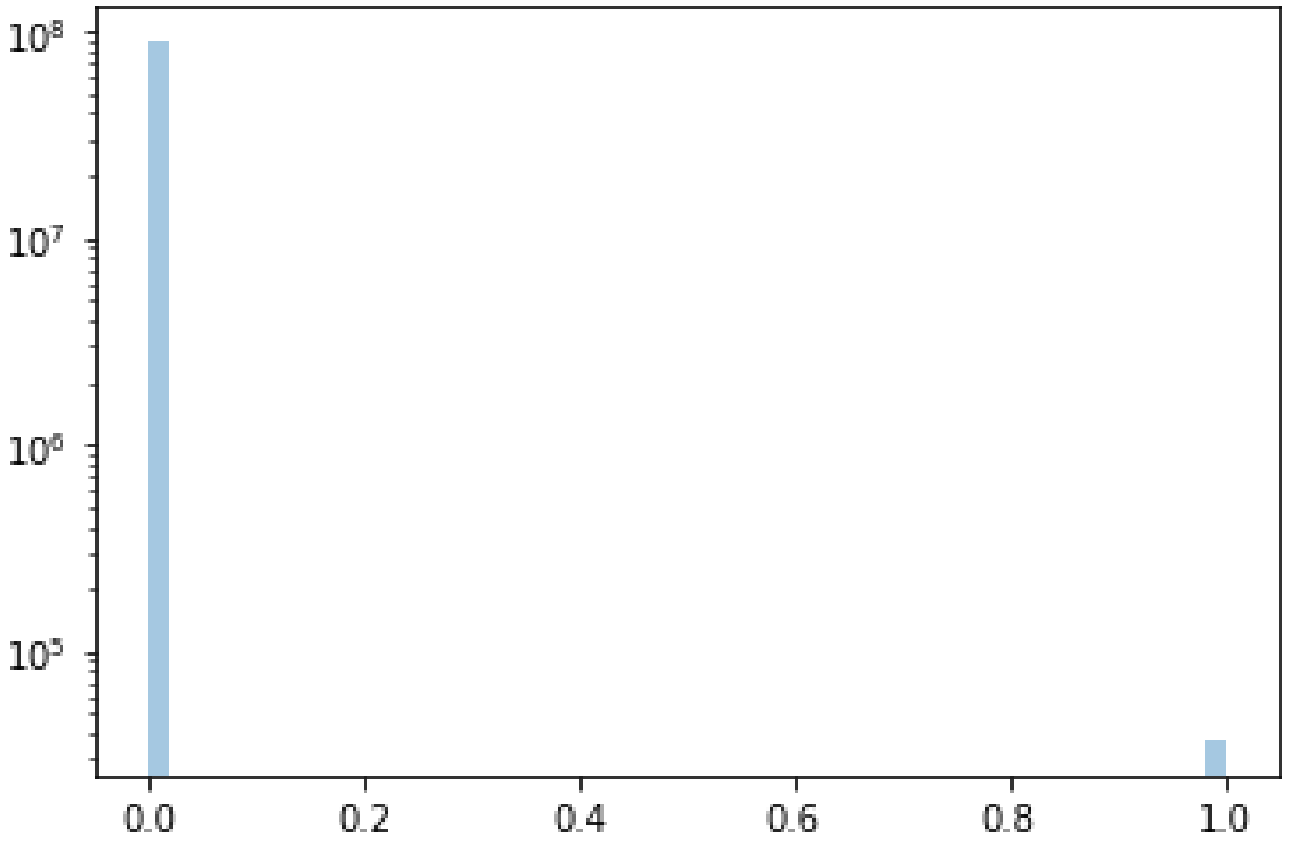
```
# waldo : not_waldo = 1 : 2 (0.34)
gen_mix = seg_gen_mix(waldo_sub_imgs, waldo_sub_labels, imgs, labels, tot_bs=4, prop=0.34, out_sz=(PANNEL_SIZE, PANNEL_SIZE))

X, y = next(gen_mix)

plt.figure(figsize=(5, 10))
for i, img in enumerate(X):
    plt.subplot(X.shape[0], 2, 2*i+1)
    plt.imshow(X[i])
    plt.subplot(X.shape[0], 2, 2*i+2)
    plt.colorbar()
    plt.imshow(y[i][:,1].reshape((PANNEL_SIZE, PANNEL_SIZE)))
```

Preview Sample Pannel Images 이미지 샘플 확인





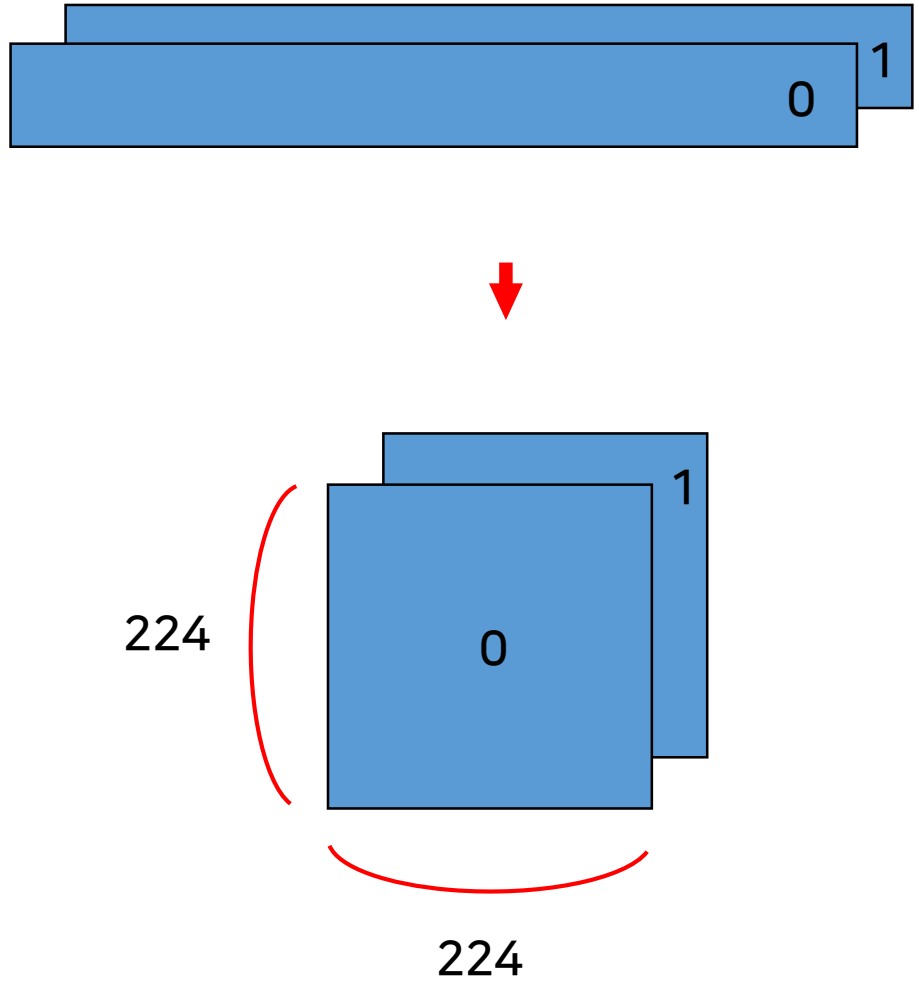
Plot Y-Data Distribution 데이터의 분포 확인



각 라벨에 대한 데이터셋의 수가 비슷해야 잘 학습된다.

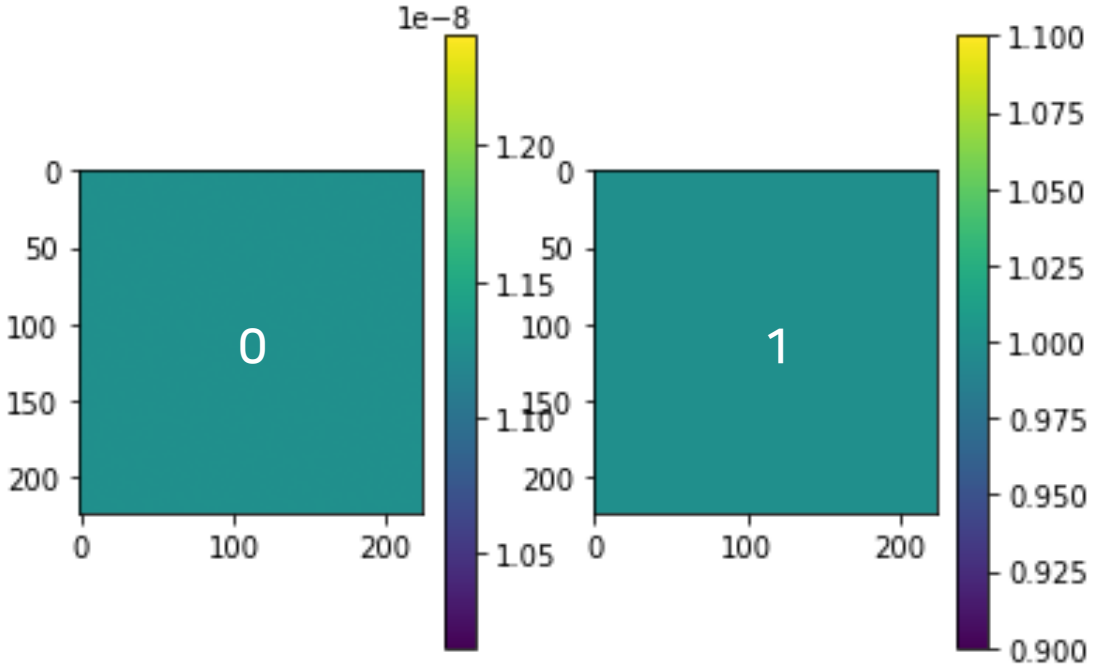
Make Class Weights 데이터의 분포 확인.

```
sample_weights = np.zeros((6, PANNEL_SIZE * PANNEL_SIZE, 2))

sample_weights[:,:,0] = 1. / freq0
sample_weights[:,:,1] = 1.
```



Make Class Weights 데이터의 분포 확인.

```
plt.subplot(1,2,1)
plt.imshow(sample_weights[0,:,0].reshape((224, 224)))
plt.colorbar()
plt.subplot(1,2,2)
plt.imshow(sample_weights[0,:,1].reshape((224, 224)))
plt.colorbar()
```



Create Model

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv2d (Conv2D)	(None, 224, 224, 64)	1792	input_1[0][0]
leaky_re_lu (LeakyReLU)	(None, 224, 224, 64)	0	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0	leaky_re_lu[0][0]
conv2d_1 (Conv2D)	(None, 112, 112, 128	73856	max_pooling2d[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 112, 112, 128	0	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0	leaky_re_lu_1[0][0]
conv2d_2 (Conv2D)	(None, 56, 56, 256)	295168	max_pooling2d_1[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 56, 56, 256)	0	conv2d_2[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0	leaky_re_lu_2[0][0]
conv2d_3 (Conv2D)	(None, 28, 28, 256)	65792	max_pooling2d_2[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 28, 28, 256)	0	conv2d_3[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0	leaky_re_lu_3[0][0]

다운 샘플링

224 ▷ 112 ▷ 56 ▷ 28 ▷ 14

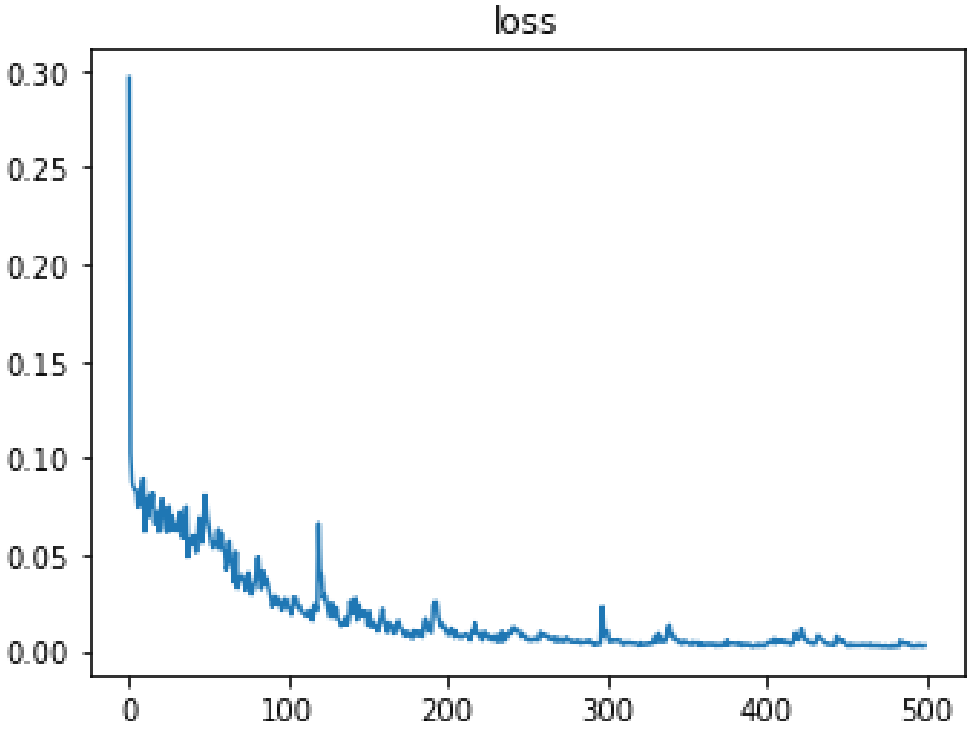
Create Model

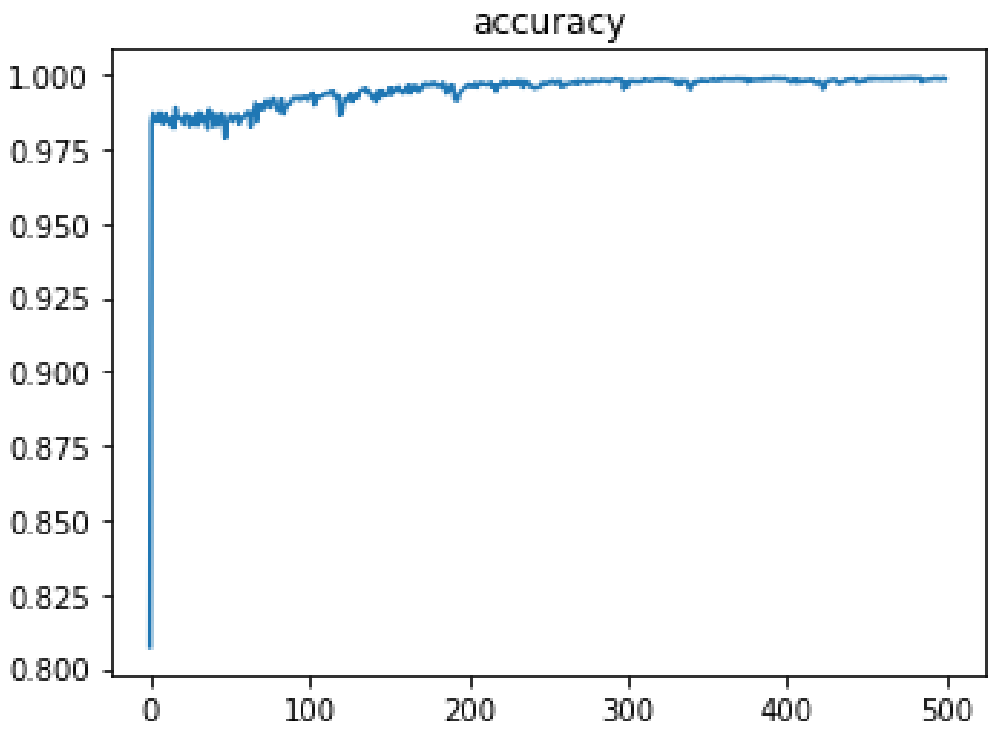
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0	leaky_re_lu_3[0][0]
up_sampling2d (UpSampling2D)	(None, 28, 28, 256)	0	max_pooling2d_3[0][0]
conv2d_4 (Conv2D)	(None, 28, 28, 256)	590080	up_sampling2d[0][0]
activation (Activation)	(None, 28, 28, 256)	0	conv2d_4[0][0]
add (Add)	(None, 28, 28, 256)	0	activation[0][0] max_pooling2d_2[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 56, 56, 256)	0	add[0][0]
conv2d_5 (Conv2D)	(None, 56, 56, 128)	295040	up_sampling2d_1[0][0]
activation_1 (Activation)	(None, 56, 56, 128)	0	conv2d_5[0][0]
add_1 (Add)	(None, 56, 56, 128)	0	activation_1[0][0] max_pooling2d_1[0][0]
up_sampling2d_2 (UpSampling2D)	(None, 112, 112, 128	0	add_1[0][0]
conv2d_6 (Conv2D)	(None, 112, 112, 64)	73792	up_sampling2d_2[0][0]
activation_2 (Activation)	(None, 112, 112, 64)	0	conv2d_6[0][0]
add_2 (Add)	(None, 112, 112, 64)	0	activation_2[0][0] max_pooling2d[0][0]
up_sampling2d_3 (UpSampling2D)	(None, 224, 224, 64)	0	add_2[0][0]
conv2d_7 (Conv2D)	(None, 224, 224, 2)	130	up_sampling2d_3[0][0]
reshape (Reshape)	(None, 50176, 2)	0	conv2d_7[0][0]
activation_3 (Activation)	(None, 50176, 2)	0	reshape[0][0]

업 샘플링

14 ▷ 28 ▷ 56 ▷ 112 ▷ 224

Train


loss
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
100
200
300
400
500

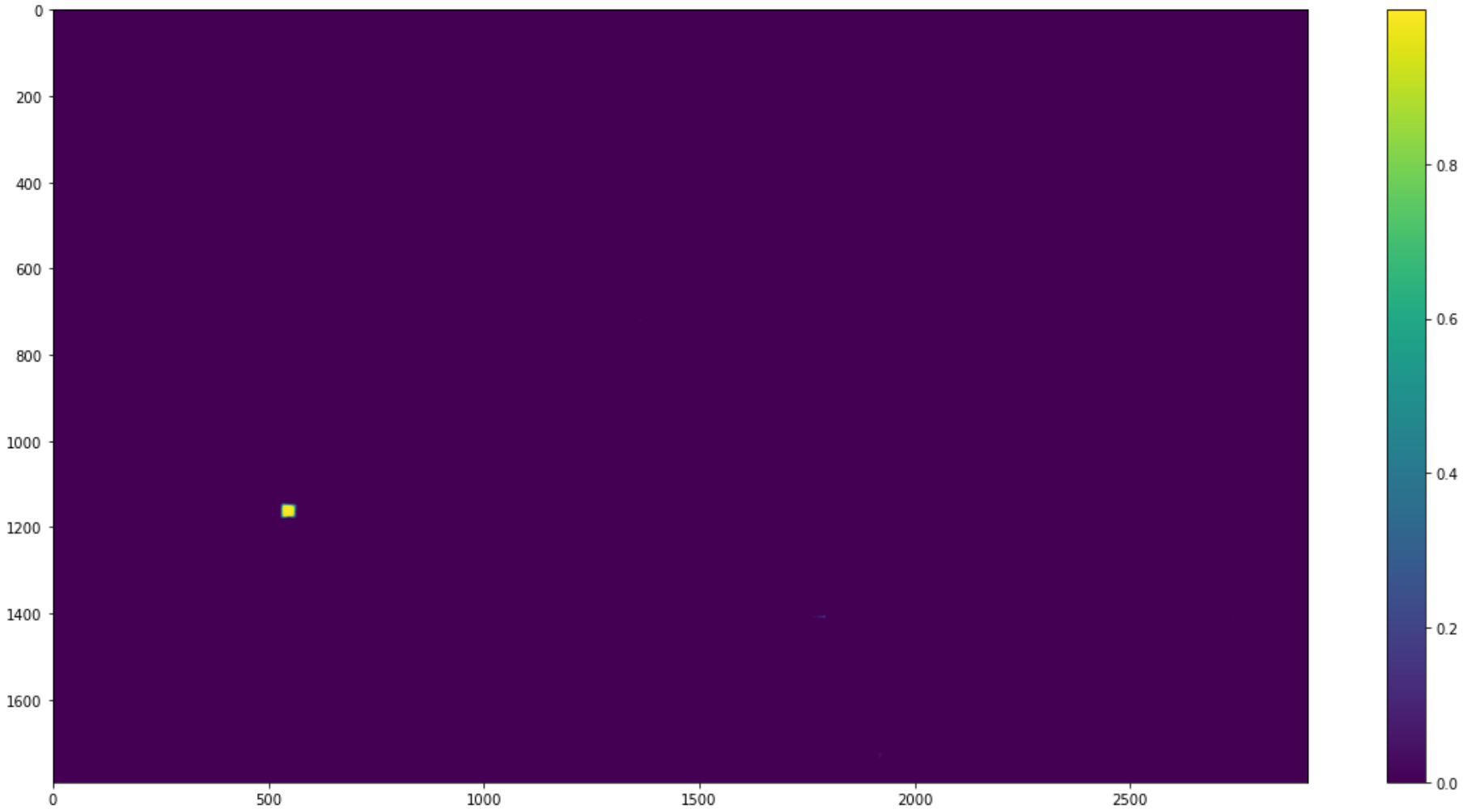

accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0
100
200
300
400
500

Evaluation

이미지 변경 가능 (ex. 01.jpg)

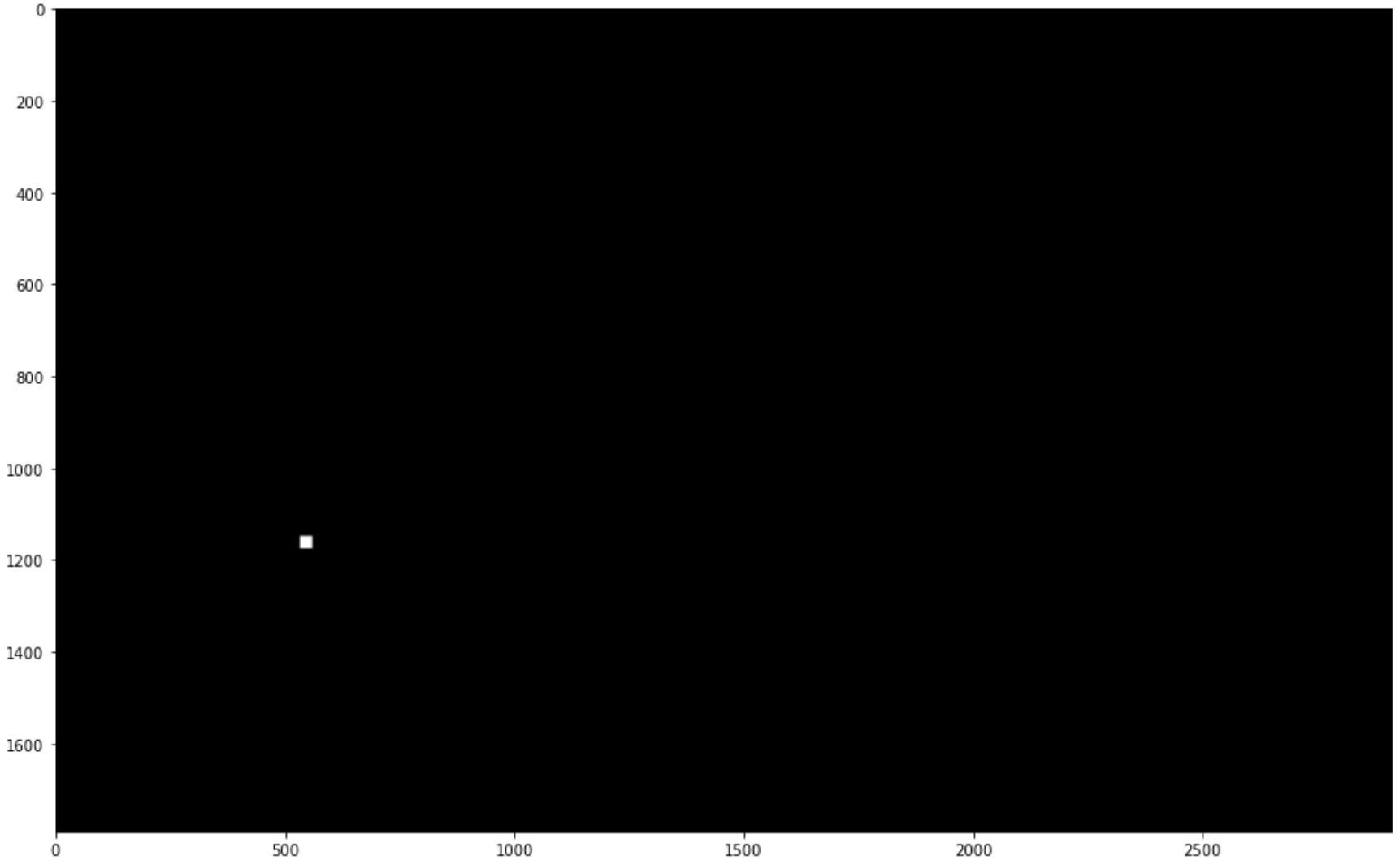
```
img_filename = '02.jpg'
test_img = np.array(Image.open(os.path.join('test_imgs', img_filename)).resize((2800, 1760), Image.NEAREST)).astype(np.float32) / 255.

plt.figure(figsize=(20, 10))
plt.imshow(test_img)
```

Predict



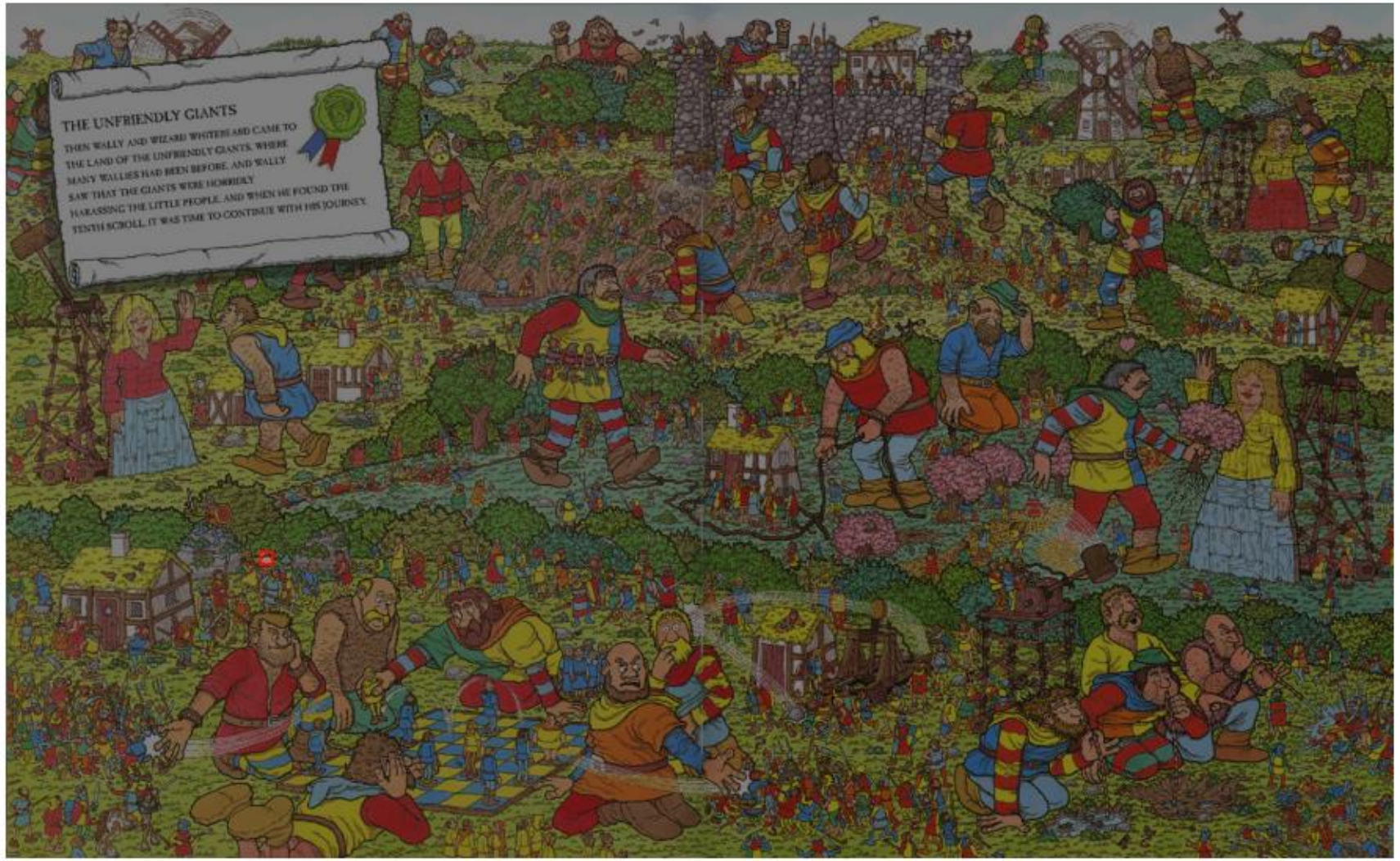
웰리가 있는 곳은 1 나머지는 0으로 표시

Make Overlay for Result



이미지에 결과를 오버레이

Final Result



월리가 있는 곳에 빨간 사각형으로 표시