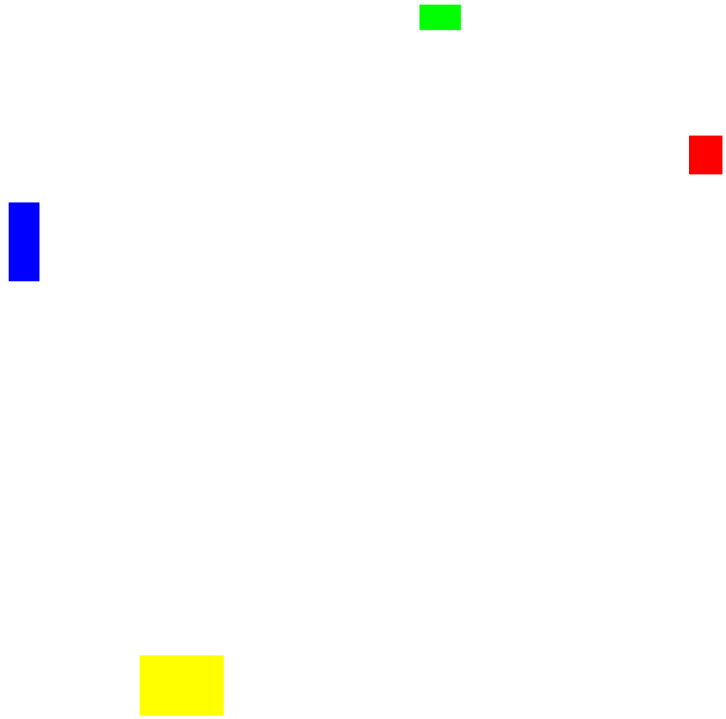
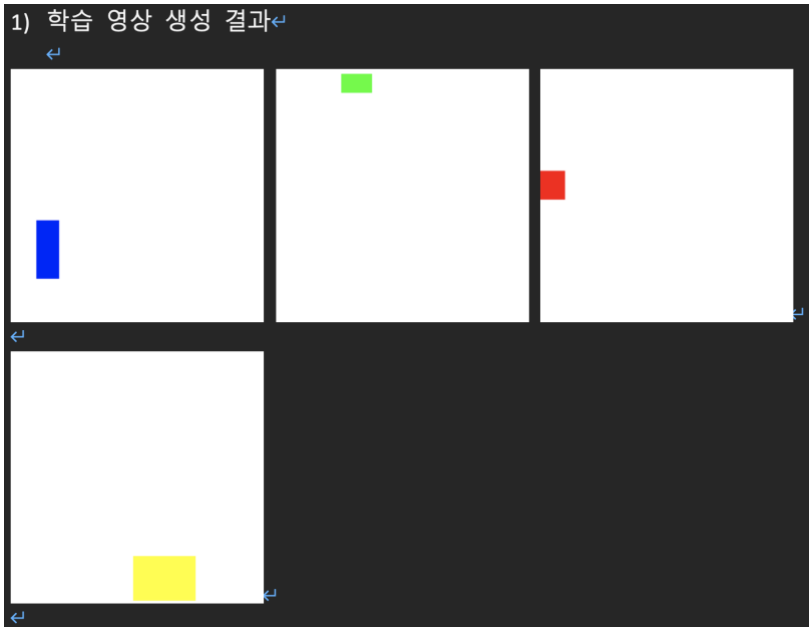
머신비전시스템 과제 7

18011789 조혜수

1) 학습 영상 생성 결과



<검정배경에서의 모습>



2) 검출기 구조

<복사>

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
Connected to
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3  0           []
                             )]

 block1_conv1 (Conv2D)       (None, 256, 256, 64  1792
['input_2[0][0]']
                             )

 block1_conv2 (Conv2D)       (None, 256, 256, 64  36928
['block1_conv1[0][0]']
                             )

 block1_pool (MaxPooling2D)  (None, 128, 128, 64  0
['block1_conv2[0][0]']
                             )

 block2_conv1 (Conv2D)       (None, 128, 128, 12  73856
['block1_pool[0][0]']
                            8)

 block2_conv2 (Conv2D)       (None, 128, 128, 12  147584
['block2_conv1[0][0]']
                            8)

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)  0
['block2_conv2[0][0]']

 block3_conv1 (Conv2D)       (None, 64, 64, 256)  295168
['block2_pool[0][0]']

 block3_conv2 (Conv2D)       (None, 64, 64, 256)  590080
['block3_conv1[0][0]']

 block3_conv3 (Conv2D)       (None, 64, 64, 256)  590080
['block3_conv2[0][0]']

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)  0
['block3_conv3[0][0]']

 block4_conv1 (Conv2D)       (None, 32, 32, 512)  1180160
['block3_pool[0][0]']

 block4_conv2 (Conv2D)       (None, 32, 32, 512)  2359808
['block4_conv1[0][0]']

 block4_conv3 (Conv2D)       (None, 32, 32, 512)  2359808
['block4_conv2[0][0]']
```

```
 block4_pool (MaxPooling2D)     (None, 16, 16, 512)  0
['block4_conv3[0][0]']

 block5_conv1 (Conv2D)          (None, 16, 16, 512)  2359808
['block4_pool[0][0]']

 block5_conv2 (Conv2D)          (None, 16, 16, 512)  2359808
['block5_conv1[0][0]']

 block5_conv3 (Conv2D)          (None, 16, 16, 512)  2359808
['block5_conv2[0][0]']

 block5_pool (MaxPooling2D)     (None, 8, 8, 512)    0
['block5_conv3[0][0]']

 flatten_1 (Flatten)            (None, 32768)        0
['block5_pool[0][0]']

 out_coord (Dense)              (None, 2)            65538
['flatten_1[0][0]']

 out_size (Dense)               (None, 2)            65538
['flatten_1[0][0]']

 out_class (Dense)              (None, 4)            131076
['flatten_1[0][0]']

==================================================================================
==========================
Total params: 14,976,840
Trainable params: 14,976,840
Non-trainable params: 0
__________________________________________________________________________________
__________________________
```

<캡쳐>

```
Model: "model_1"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3       0           []
                             )]

 block1_conv1 (Conv2D)       (None, 256, 256, 64       1792        ['input_2[0][0]']
                             )

 block1_conv2 (Conv2D)       (None, 256, 256, 64       36928       ['block1_conv1[0][0]']
                             )

 block1_pool (MaxPooling2D)  (None, 128, 128, 64       0           ['block1_conv2[0][0]']
                             )

 block2_conv1 (Conv2D)       (None, 128, 128, 12       73856       ['block1_pool[0][0]']
                             8)

 block2_conv2 (Conv2D)       (None, 128, 128, 12       147584      ['block2_conv1[0][0]']
                             8)

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0           ['block2_conv2[0][0]']

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168      ['block2_pool[0][0]']

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080      ['block3_conv1[0][0]']

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080      ['block3_conv2[0][0]']

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0           ['block3_conv3[0][0]']

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160     ['block3_pool[0][0]']

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808     ['block4_conv1[0][0]']

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808     ['block4_conv2[0][0]']

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0           ['block4_conv3[0][0]']

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808     ['block4_pool[0][0]']

block5_conv2 (Conv2D)          (None, 16, 16, 512)    2359808     ['block5_conv1[0][0]']

block5_conv3 (Conv2D)          (None, 16, 16, 512)    2359808     ['block5_conv2[0][0]']

block5_pool (MaxPooling2D)     (None, 8, 8, 512)      0           ['block5_conv3[0][0]']

flatten_1 (Flatten)            (None, 32768)          0           ['block5_pool[0][0]']

out_coord (Dense)              (None, 2)              65538       ['flatten_1[0][0]']

out_size (Dense)               (None, 2)              65538       ['flatten_1[0][0]']

out_class (Dense)              (None, 4)              131076      ['flatten_1[0][0]']

=================================================================================================
Total params: 14,976,840
Trainable params: 14,976,840
Non-trainable params: 0
_________________________________________________________________________________________________
```

3) Training Loss, Validation Loss

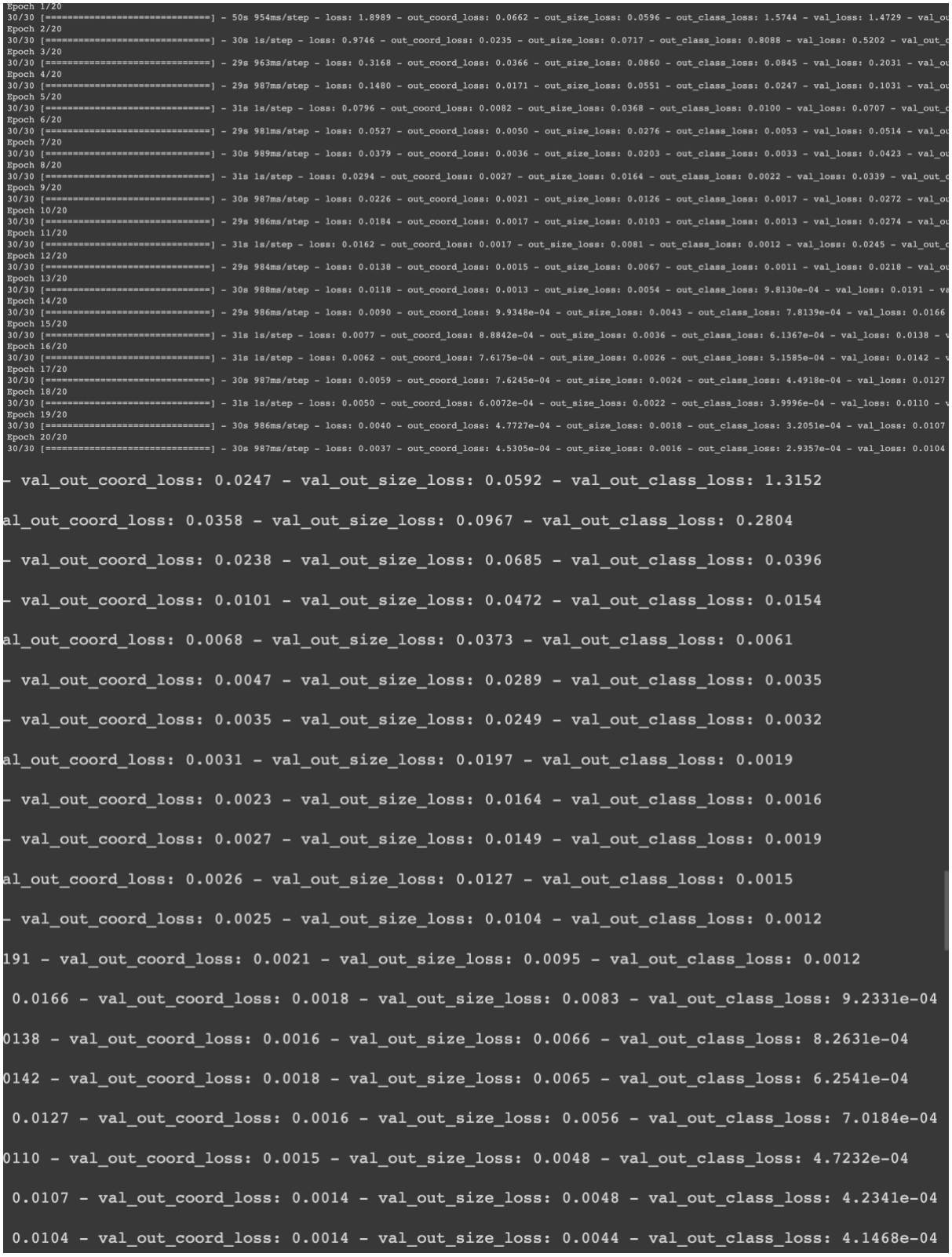
<복사>

```
Epoch 1/20
30/30 [==============================] - 50s 954ms/step - loss: 1.8989
- out_coord_loss: 0.0662 - out_size_loss: 0.0596 - out_class_loss:
```

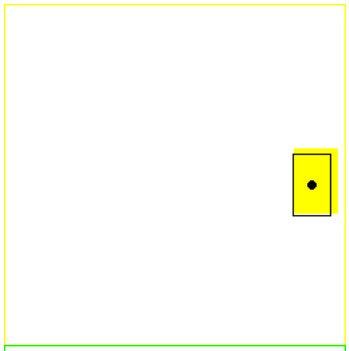
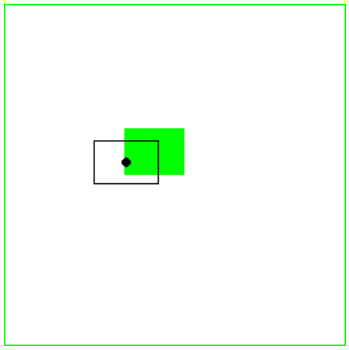
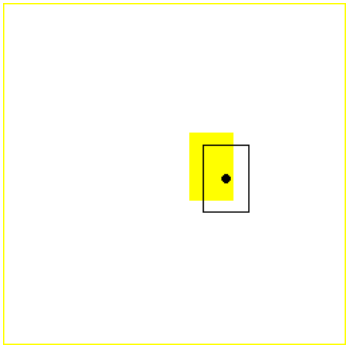
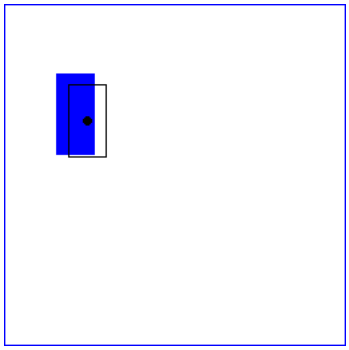
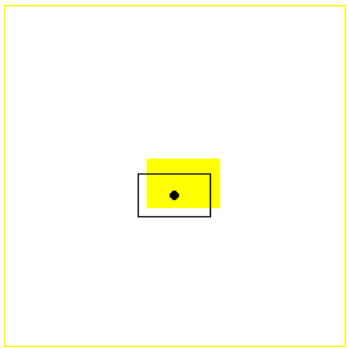
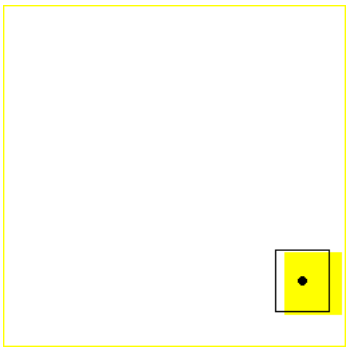
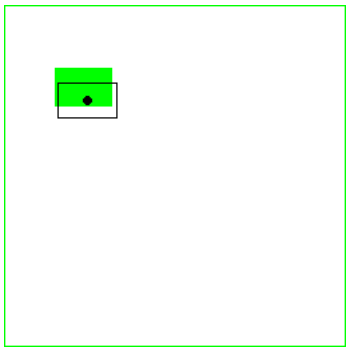
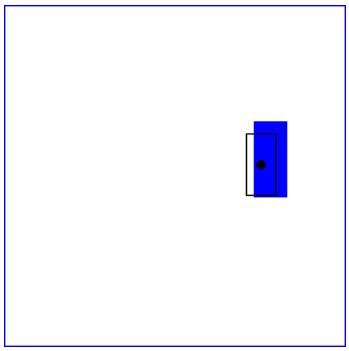
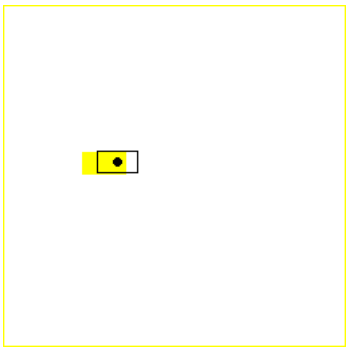
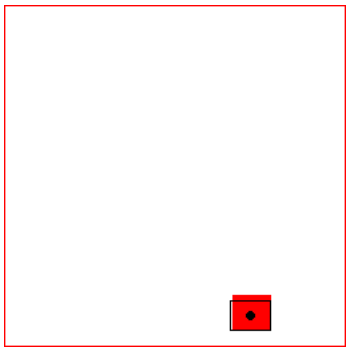
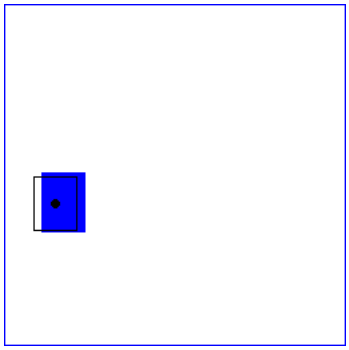
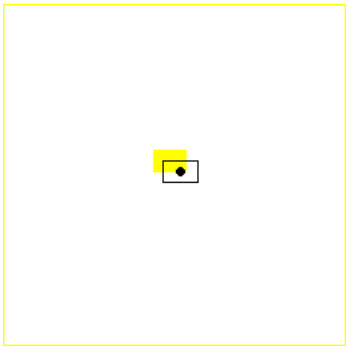
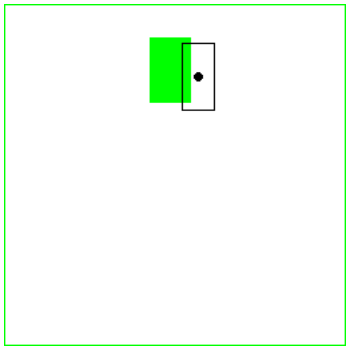
```
1.5744 - val_loss: 1.4729 - val_out_coord_loss: 0.0247 -
val_out_size_loss: 0.0592 - val_out_class_loss: 1.3152
Epoch 2/20
30/30 [==============================] - 30s 1s/step - loss: 0.9746 -
out_coord_loss: 0.0235 - out_size_loss: 0.0717 - out_class_loss: 0.8088
- val_loss: 0.5202 - val_out_coord_loss: 0.0358 - val_out_size_loss:
0.0967 - val_out_class_loss: 0.2804
Epoch 3/20
30/30 [==============================] - 29s 963ms/step - loss: 0.3168
- out_coord_loss: 0.0366 - out_size_loss: 0.0860 - out_class_loss:
0.0845 - val_loss: 0.2031 - val_out_coord_loss: 0.0238 -
val_out_size_loss: 0.0685 - val_out_class_loss: 0.0396
Epoch 4/20
30/30 [==============================] - 29s 987ms/step - loss: 0.1480
- out_coord_loss: 0.0171 - out_size_loss: 0.0551 - out_class_loss:
0.0247 - val_loss: 0.1031 - val_out_coord_loss: 0.0101 -
val_out_size_loss: 0.0472 - val_out_class_loss: 0.0154
Epoch 5/20
30/30 [==============================] - 31s 1s/step - loss: 0.0796 -
out_coord_loss: 0.0082 - out_size_loss: 0.0368 - out_class_loss: 0.0100
- val_loss: 0.0707 - val_out_coord_loss: 0.0068 - val_out_size_loss:
0.0373 - val_out_class_loss: 0.0061
Epoch 6/20
30/30 [==============================] - 29s 981ms/step - loss: 0.0527
- out_coord_loss: 0.0050 - out_size_loss: 0.0276 - out_class_loss:
0.0053 - val_loss: 0.0514 - val_out_coord_loss: 0.0047 -
val_out_size_loss: 0.0289 - val_out_class_loss: 0.0035
Epoch 7/20
30/30 [==============================] - 30s 989ms/step - loss: 0.0379
- out_coord_loss: 0.0036 - out_size_loss: 0.0203 - out_class_loss:
0.0033 - val_loss: 0.0423 - val_out_coord_loss: 0.0035 -
val_out_size_loss: 0.0249 - val_out_class_loss: 0.0032
Epoch 8/20
30/30 [==============================] - 31s 1s/step - loss: 0.0294 -
out_coord_loss: 0.0027 - out_size_loss: 0.0164 - out_class_loss: 0.0022
- val_loss: 0.0339 - val_out_coord_loss: 0.0031 - val_out_size_loss:
0.0197 - val_out_class_loss: 0.0019
Epoch 9/20
30/30 [==============================] - 30s 987ms/step - loss: 0.0226
- out_coord_loss: 0.0021 - out_size_loss: 0.0126 - out_class_loss:
0.0017 - val_loss: 0.0272 - val_out_coord_loss: 0.0023 -
val_out_size_loss: 0.0164 - val_out_class_loss: 0.0016
Epoch 10/20
30/30 [==============================] - 29s 986ms/step - loss: 0.0184
- out_coord_loss: 0.0017 - out_size_loss: 0.0103 - out_class_loss:
0.0013 - val_loss: 0.0274 - val_out_coord_loss: 0.0027 -
val_out_size_loss: 0.0149 - val_out_class_loss: 0.0019
Epoch 11/20
30/30 [==============================] - 31s 1s/step - loss: 0.0162 -
out_coord_loss: 0.0017 - out_size_loss: 0.0081 - out_class_loss: 0.0012
- val_loss: 0.0245 - val_out_coord_loss: 0.0026 - val_out_size_loss:
0.0127 - val_out_class_loss: 0.0015
Epoch 12/20
30/30 [==============================] - 29s 984ms/step - loss: 0.0138
- out_coord_loss: 0.0015 - out_size_loss: 0.0067 - out_class_loss:
0.0011 - val_loss: 0.0218 - val_out_coord_loss: 0.0025 -
val_out_size_loss: 0.0104 - val_out_class_loss: 0.0012
Epoch 13/20
```

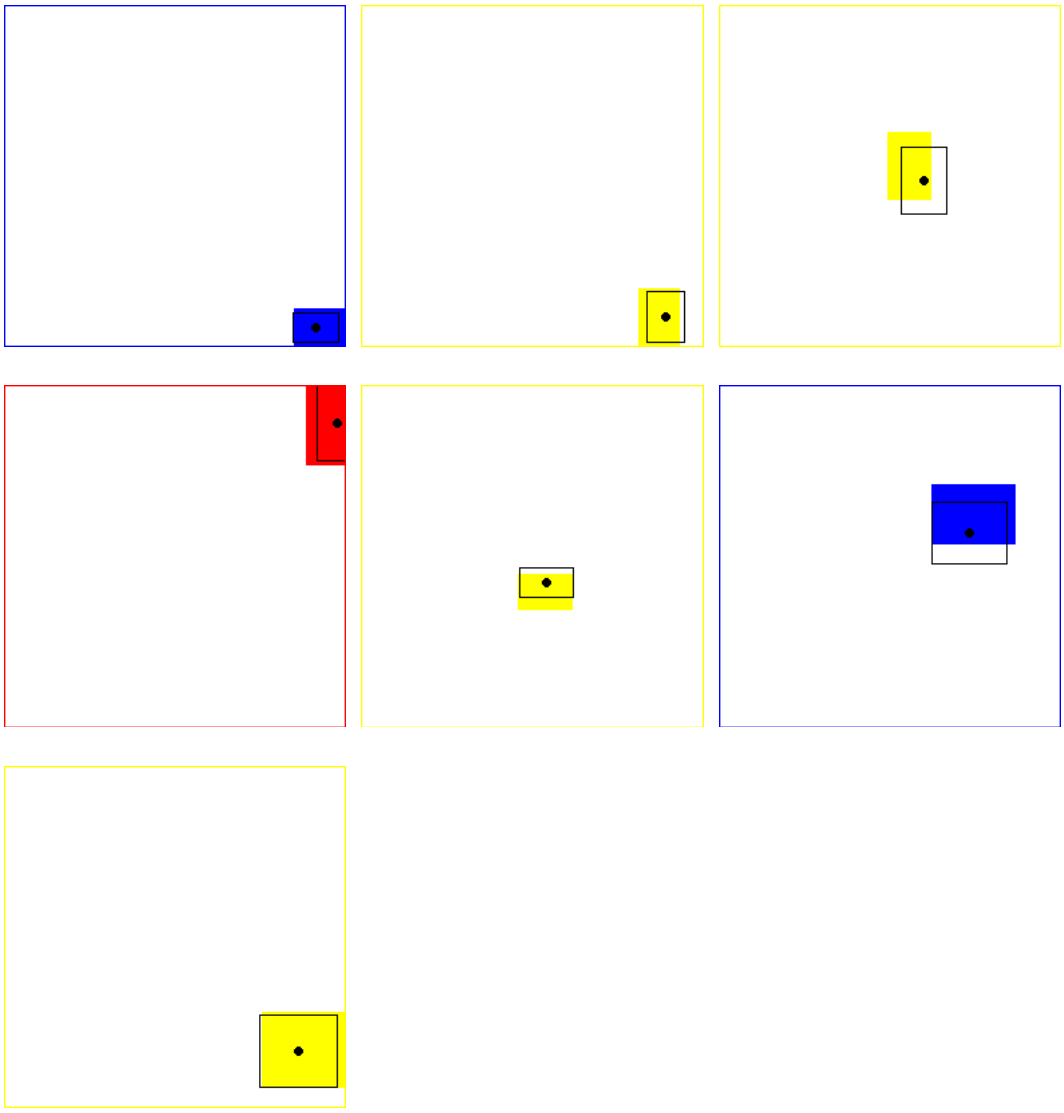
```
30/30 [==============================] - 30s 988ms/step - loss: 0.0118
- out_coord_loss: 0.0013 - out_size_loss: 0.0054 - out_class_loss:
9.8130e-04 - val_loss: 0.0191 - val_out_coord_loss: 0.0021 -
val_out_size_loss: 0.0095 - val_out_class_loss: 0.0012
Epoch 14/20
30/30 [==============================] - 29s 986ms/step - loss: 0.0090
- out_coord_loss: 9.9348e-04 - out_size_loss: 0.0043 - out_class_loss:
7.8139e-04 - val_loss: 0.0166 - val_out_coord_loss: 0.0018 -
val_out_size_loss: 0.0083 - val_out_class_loss: 9.2331e-04
Epoch 15/20
30/30 [==============================] - 31s 1s/step - loss: 0.0077 -
out_coord_loss: 8.8842e-04 - out_size_loss: 0.0036 - out_class_loss:
6.1367e-04 - val_loss: 0.0138 - val_out_coord_loss: 0.0016 -
val_out_size_loss: 0.0066 - val_out_class_loss: 8.2631e-04
Epoch 16/20
30/30 [==============================] - 31s 1s/step - loss: 0.0062 -
out_coord_loss: 7.6175e-04 - out_size_loss: 0.0026 - out_class_loss:
5.1585e-04 - val_loss: 0.0142 - val_out_coord_loss: 0.0018 -
val_out_size_loss: 0.0065 - val_out_class_loss: 6.2541e-04
Epoch 17/20
30/30 [==============================] - 30s 987ms/step - loss: 0.0059
- out_coord_loss: 7.6245e-04 - out_size_loss: 0.0024 - out_class_loss:
4.4918e-04 - val_loss: 0.0127 - val_out_coord_loss: 0.0016 -
val_out_size_loss: 0.0056 - val_out_class_loss: 7.0184e-04
Epoch 18/20
30/30 [==============================] - 31s 1s/step - loss: 0.0050 -
out_coord_loss: 6.0072e-04 - out_size_loss: 0.0022 - out_class_loss:
3.9996e-04 - val_loss: 0.0110 - val_out_coord_loss: 0.0015 -
val_out_size_loss: 0.0048 - val_out_class_loss: 4.7232e-04
Epoch 19/20
30/30 [==============================] - 30s 986ms/step - loss: 0.0040
- out_coord_loss: 4.7727e-04 - out_size_loss: 0.0018 - out_class_loss:
3.2051e-04 - val_loss: 0.0107 - val_out_coord_loss: 0.0014 -
val_out_size_loss: 0.0048 - val_out_class_loss: 4.2341e-04
Epoch 20/20
30/30 [==============================] - 30s 987ms/step - loss: 0.0037
- out_coord_loss: 4.5305e-04 - out_size_loss: 0.0016 - out_class_loss:
2.9357e-04 - val_loss: 0.0104 - val_out_coord_loss: 0.0014 -
val_out_size_loss: 0.0044 - val_out_class_loss: 4.1468e-04
INFO:tensorflow:Assets written to: model/assets
```

<캡쳐>



4) 물체 검출 결과 16개 이상 (20개 첨부)



5) 코드

```python
# Import libraries
import numpy as np
import tensorflow as tf
import cv2 as cv
import random
from google.colab.patches import cv_imshow
import matplotlib.pyplot as plt
```

```python
[18] # Generate training images and labels
     N=2000
     H,W=256,256
     train_img=np.zeros([N,H,W,3],dtype=np.uint8)
     train_img.fill(255)
     train_label_coord=np.zeros([N,2],dtype=np.int32)
     train_label_size=np.zeros([N,2],dtype=np.int32)
     train_label_class=np.zeros([N,1],dtype=np.int32) #color
```

```python
[19] for n in range(N):
         x,y=random.randint(0,W-1),random.randint(0,H-1)
         bw,bh=random.randint(int(W/16),int(W/4)),random.randint(int(H/16),int(H/4))

         if(x-bw/2<0): x=x-(x-bw/2)
         elif(x+bw/2>W-1): x=x-(x+bw/2-(W-1))
         if(y-bh/2<0): y=y-(y-bh/2)
         elif(y+bh/2>H-1): y=y-(y+bh/2-(H-1))

         x=int(x); y=int(y)
         train_label_coord[n,0]=x; train_label_coord[n,1]=y
         train_label_size[n,0]=bw; train_label_size[n,1]=bh
         train_label_class[n]=random.randint(0,3) # 0: red, 1: green, 2: blue, 3: yellow

         if train_label_class[n]==0: # red
             cv.rectangle(train_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(0,0,255), thickness=-1)
         elif train_label_class[n]==1: # green
             cv.rectangle(train_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(0,255,0), thickness=-1)
         elif train_label_class[n]==2: # blue
             cv.rectangle(train_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(255,0,0), thickness=-1)
         else: # yellow
             cv.rectangle(train_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(0,255,255), thickness=-1)
```

```python
[20] # Display some images
     for n in range(8):
         cv_imshow(train_img[n])
```

```
[21] # Preprocess data
     train_img = tf.keras.applications.vgg16.preprocess_input(train_img)

     train_label_coord=train_label_coord.astype(np.float32)
     train_label_coord[:,0]=train_label_coord[:,0]/W
     train_label_coord[:,1]=train_label_coord[:,1]/H
     train_label_size=train_label_size.astype(np.float32)
     train_label_size[:,0]=train_label_size[:,0]/(W/4)
     train_label_size[:,1]=train_label_size[:,1]/(H/4)
```

```
[22] # Build model
     base_model = tf.keras.applications.VGG16(input_shape=[H,W,3], include_top=False, weights='imagenet')
     x = base_model.output
     x = tf.keras.layers.Flatten()(x)
     out_coord=tf.keras.layers.Dense(2,activation='sigmoid',name='out_coord')(x)
     out_size=tf.keras.layers.Dense(2, activation='sigmoid',name='out_size')(x)
     out_class=tf.keras.layers.Dense(4,activation='softmax',name='out_class')(x)

     model=tf.keras.Model(inputs=base_model.input,outputs=[out_coord,out_size,out_class])
     model.summary()
```

```
Model: "model_1"
```

```
[23] # Custom loss functions
     def coord_loss_func(y_true,y_pred):
         loss_coord=tf.keras.losses.mean_squared_error(y_true,y_pred)
         return loss_coord
     def size_loss_func(y_true,y_pred):
         loss_size=tf.keras.losses.mean_squared_error(y_true,y_pred)
         return loss_size
     def class_loss_func(y_true,y_pred):
         loss_class=tf.keras.losses.sparse_categorical_crossentropy(y_true,y_pred)
         return loss_class
```

```
[24] # Train and save model
     model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                   loss={'out_coord':coord_loss_func,'out_size':size_loss_func,'out_class':class_loss_func},
                   loss_weights={'out_coord':4,'out_size':1,'out_class':1})
```

```
model.summary()
```

```
[26] history = model.fit(x=train_img,
                         y={'out_coord':train_label_coord,'out_size':train_label_size,'out_class':train_label_class},
                         epochs=20,batch_size=50,validation_split=0.25)
     model.save('model')
```

```
[27] # Generate test images and labels
     N=20
     H,W=256,256
     test_img=np.zeros([N,H,W,3],dtype=np.uint8)
     test_img.fill(255)
     test_label_coord=np.zeros([N,2],dtype=np.int32)
     test_label_size=np.zeros([N,2],dtype=np.int32)
     test_label_class=np.zeros([N,1],dtype=np.int32)
```

```
[28] for n in range(N):
         x,y=random.randint(0,W-1),random.randint(0,H-1)
         bw,bh=random.randint(int(W/16),int(W/4)),random.randint(int(H/16),int(H/4))

         if(x-bw/2<0): x=x-(x-bw/2)
         elif(x+bw/2>W-1):  x=x-(x+bw/2-(W-1))
         if(y-bh/2<0):  y=y-(y-bh/2)
         elif(y+bh/2>H-1): y=y-(y+bh/2-(H-1))

         x=int(x); y=int(y)
         test_label_coord[n,0]=x; test_label_coord[n,1]=y
         test_label_size[n,0]=bw; test_label_size[n,1]=bh
         test_label_class[n]=random.randint(0,3) # color

         if test_label_class[n]==0:
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(0,0,255),thickness=-1)
         elif train_label_class[n]==1:
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(0,255,0),thickness=-1)
         elif train_label_class[n]==2:
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(255,0,0),thickness=-1)
         else:
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)), color=(0,255,255),thickness=-1)
```

```
[29] # Preprocess test images
     test_img_ = tf.keras.applications.vgg16.preprocess_input(test_img)
     # Predict object locations in test images
     model=tf.keras.models.load_model('model',
                                      custom_objects={'coord_loss_func':coord_loss_func,
                                                      'size_loss_func':size_loss_func,
                                                      'class_loss_func':class_loss_func})

     pred_coord,pred_size,pred_class=model.predict(test_img_)
     pred_coord[:,0]=pred_coord[:,0]*H
     pred_coord[:,1]=pred_coord[:,1]*W
     pred_size[:,0]=pred_size[:,0]*(H/4)
     pred_size[:,1]=pred_size[:,1]*(W/4)
     pred_class=np.argmax(pred_class,axis=1)
```

```
[30] # Display prediction results
     for n in range(N):
         x=pred_coord[n,0].astype('int')
         y=pred_coord[n,1].astype('int')
         bw=pred_size[n,0].astype('int')
         bh=pred_size[n,1].astype('int')
         obj_class=pred_class[n]

         if obj_class==0:
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)),color=(0,0,0),thickness=1)
             cv.rectangle(test_img[n],(0,0),(W-1,H-1),color=(0,0,255),thickness=1)

         elif obj_class==1:
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)),color=(0,0,0),thickness=1)
             cv.rectangle(test_img[n],(0,0),(W-1,H-1),color=(0,255,0),thickness=1)
         elif obj_class==2:
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)),color=(0,0,0),thickness=1)
             cv.rectangle(test_img[n],(0,0),(W-1,H-1),color=(255,0,0),thickness=1)
         else: # ellipse
             cv.rectangle(test_img[n],(x-int(bw/2),y-int(bh/2)),(x+int(bw/2),y+int(bh/2)),color=(0,0,0),thickness=1)
             cv.rectangle(test_img[n],(0,0),(W-1,H-1),color=(0,255,255),thickness=1)

         cv.circle(test_img[n],center=(x,y),radius=2,color=(0,0,0),thickness=2)
         cv_imshow(test_img[n])
```