MobileNet V2 SSD Tensorflow로 구현해보기 실습

지난 실습 때 MobileNet V2를 이용하여 이미지 분류를 하는 모델을 학습하였다. 이번 실습에서는 조금 더 나아가 이미지 안의 물체를 탐지하는 네트워크를 실습해 볼 것이다. 지난 시간에 다뤘던 MobileNet V2에 SSD(Single Shot Detection)를 추가해 물체 탐지를 해보자.

In [1]:

```
#pip install -U tensorflow==2.5.0
```

In [2]:

```
import os
import socket
import pickle
import time
import numpy as np
import cv2
import struct
from tqdm import tqdm
from sys import getsizeof
from datetime import datetime
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Add, ReLU, Input, Dense, Dropout, Activation
    , Conv2D, MaxPooling2D, InputLayer, Reshape, DepthwiseConv2D, BatchNormaliza
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.preprocessing.image import ImageDataGenerator
#from keras.engine.topology import Input
from tensorflow.keras.optimizers import RMSprop

from tensorflow.keras.callbacks import CSVLogger
```

In [3]:

```
from tensorflow.python.keras import backend
from tensorflow.python.keras.utils import layer_utils
from tensorflow.python.keras.applications import imagenet_utils
```

In [4]:

```
from sklearn import model_selection
import math
```

Setting GPU

GPU가 없으면 아래 Step은 건너뛰어도 좋다.

In [5]:

```
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_visible_devices(gpu, 'GPU')
    tf.config.experimental.set_memory_growth(gpu, True)
```

Setting Hyperparameters

SSD의 주요 Parameter들을 아래와 같이 설정해 주자.

In [6]:

```
IMG_SIZE = 224
n_classes = 10
pos_iou_threshold = 0.3
neg_iou_threshold = 0.5
score_threshold = 0.01
layer_width=[14,7,4,2,1]
num_boxes = [3,3,3,3,3]
aspect_ratio = [1,2,1/2]
s_max = 0.9
s_min = 0.5
batch_size = 32
log_dir = './'
model_name = 'mobilenetSSD'
model_csv_path  = os.path.join(log_dir, (model_name + '.csv'))
```

Dataset

데이터셋은 cifar 10을 이용할 것이다. 다만, cifar 10은 이미지분류를 위한 데이터셋으로 Object detection 모델을 훈련시키기에는 적합하지 않으므로, 이를 적절하게 변형해주는 작업을 추가로 수행할 것이다. 먼저 데이터셋을 로드한다.

In [7]:

```
#Load data
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

train_size = x_train.shape[0]
test_size = x_test.shape[0]
```

아래는 SSD 데이터 전처리를 위한 Utility function들이다.

1. calc_iou
   두 Bounding box를 Input으로 받아 IoU(Intersection over Union)을 계산한다. 입력된 array의 마지막 dimension의 마지막 4자리를 Bounding box로 보고, 입력 포맷은 Corner 스타일(xmin, ymin, xmax, ymax) 이어야 한다.
2. match_bipartite_greedy
   각 Ground Truth별로 가장 많이 겹치는 Anchor box를 찾기 위한 함수. (Ground Truth) x (Anchor boxes) 형태의 IoU Matrix를 입력으로 받아 각 Ground Truth마다 가장 IoU가 높은 Anchor box를 찾아준다.
3. match_multi
   match_bipartite_greedy에서는 해당되지 않지만, IoU가 높은 Anchor box들에 대해서도 Ground Truth를 매칭시켜주기 위한 함수. 각 Anchor box별로 iou가 특정 임계값보다 높은 Ground Truth 중 IoU가 가장 높은 Ground Truth를 찾아준다.

4. convert_coord
   좌표값 표현 포맷을 변환해주는 함수. Centroid(cx, cy, w, h)와 Corner(xmin, ymin, xmax, ymax) 스타일 간에 변환이 가능하다.

In [8]:
```
def calc_iou(gt, anchor_boxes):
        """
        Calculate IOU of ground truth and anchor boxes

        Input:
            gt: ground truth image, shape: (#object per image, 4)
            anchor_boxes: anchor boxes, shape: (sum of grid size of all classifi
        Output:
            Matrix of iou. Row indicates each ground truth box and column indica
            shape: (#object per image, sum of grid size of all classifier)
        """

        m = gt.shape[0] # Object per image
        n = anchor_boxes.shape[0] # Number of all boxes

        #Calculate min_xy
        min_xy = np.maximum(np.tile(np.expand_dims(gt[:,0:2], axis = 1), reps =
                            np.tile(np.expand_dims(anchor_boxes[:, 0:2], axis =

        #Calculate max_xy
        max_xy = np.minimum(np.tile(np.expand_dims(gt[:,2:4], axis = 1), reps =
                            np.tile(np.expand_dims(anchor_boxes[:, 2:4], axis =

        #calculate intersection
        intersection = np.maximum((max_xy - min_xy)[:,:,0],0) * np.maximum((max_

        #calculate union
        edge_gt = np.tile(np.expand_dims(gt[:,2:4] - gt[:,0:2], axis = 1), reps
        area_gt = edge_gt[:,:,0] * edge_gt[:,:,1]

        edge_anchor_boxes = np.tile(np.expand_dims(anchor_boxes[:,2:4] - anchor_
        area_anchor_boxes = edge_anchor_boxes[:,:,0] * edge_anchor_boxes[:,:,1]

        union = area_gt + area_anchor_boxes - intersection

        return intersection / union
```

In [9]:
```
def match_bipartite_greedy(weight_matrix):
    """
    Calculate the highest matching anchor box per each ground truth
    Input: iou between each ground truth and anchor boxes, shape: (#gt, #anchor
    Output: List of matched anchor per each ground truth
    """
    m = weight_matrix.shape[0]
    n = weight_matrix.shape[1]

    matches = np.zeros(m, dtype = np.int)
    weight_cp = np.copy(weight_matrix)

    #Find the largest iou per each ground truth box in descending order
    for _ in range(m):
        largest_indices = np.argmax(weight_cp, axis = 1)
        iou_largest = weight_cp[list(range(m)), largest_indices]
        match_gt = np.argmax(iou_largest, axis = 0)
```

```
        match_anchor = largest_indices[match_gt]
        matches[match_gt] = match_anchor

        #Set the selected ground truth to 0, matched anchor box to 0 as well.
        weight_cp[match_gt, :] = 0
        weight_cp[:, match_anchor] = 0

    return matches
```

In [10]:

```
def match_multi(weight_matrix, threshold):
    """
    Multiple object match
    From remaining anchor boxes, find the most similar ground truth
    whose iou is greater than pos_threshold
    """
    m = weight_matrix.shape[0]
    n = weight_matrix.shape[1]

    #Find the largest iou per each anchor box
    largest_indices = np.argmax(weight_matrix, axis = 0)
    iou_largest = weight_matrix[largest_indices, list(range(n))]

    #Filter iou is greater than the threshold
    matches_anchor = np.nonzero(iou_largest >= threshold)[0].astype(np.int)
    matches_gt = iou_largest[matches_anchor].astype(np.int)

    return matches_anchor, matches_gt
```

In [11]:

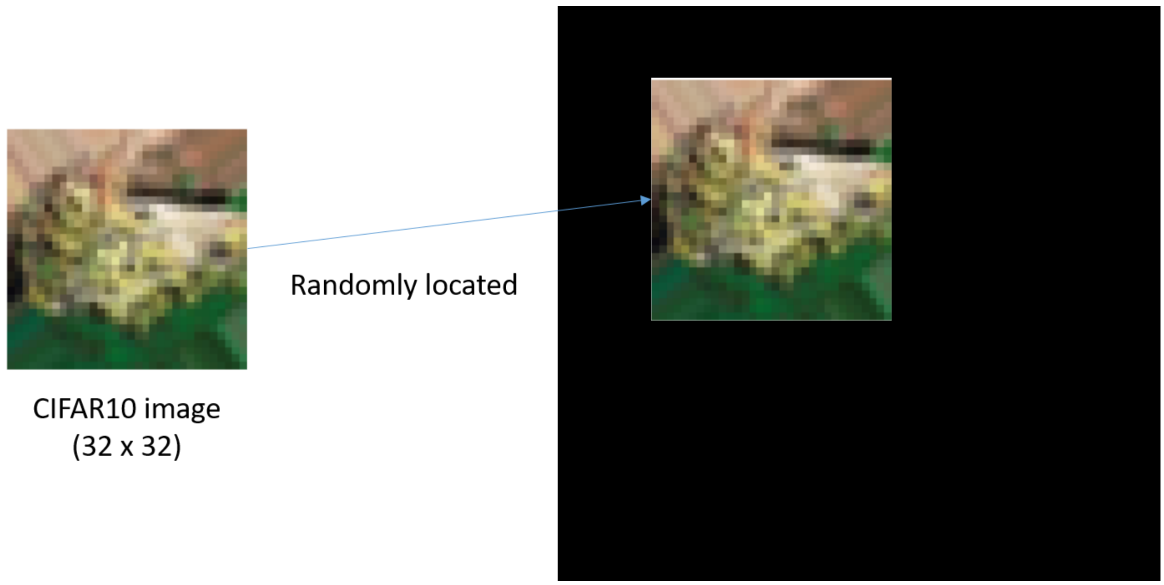
```
def convert_coord(boxes, type='centroid2corner'):
    """
        Input: Input labels
        type: how to convert
            centroid2corner: (cx, cy, w, h) -> (xmin, ymin, xmax, ymax)
            corner2centroid: (xmin, ymin, xmax, ymax) -> (cx, cy, w, h)
    """

    if type=='centroid2corner':
        cx = boxes[..., -4]
        cy = boxes[..., -3]
        w = boxes[..., -2]
        h = boxes[..., -1]

        converted_boxes = np.copy(boxes)
        converted_boxes[..., -4] = cx - w / 2 #xmin
        converted_boxes[..., -3] = cy - h / 2 #ymin
        converted_boxes[..., -2] = cx + w / 2 #xmax
        converted_boxes[..., -1] = cy + h / 2 #ymax
    elif type=='corner2centroid':
        xmin = boxes[..., -4]
        ymin = boxes[..., -3]
        xmax = boxes[..., -2]
        ymax = boxes[..., -1]

        converted_boxes = np.copy(boxes)

        converted_boxes[..., -4] = (xmin + xmax) / 2 #cx
        converted_boxes[..., -3] = (ymin + ymax) / 2 #cy
        converted_boxes[..., -2] = xmax - xmin #w
```

```
        converted_boxes[..., -1] = ymax - ymin #h

    return converted_boxes
```

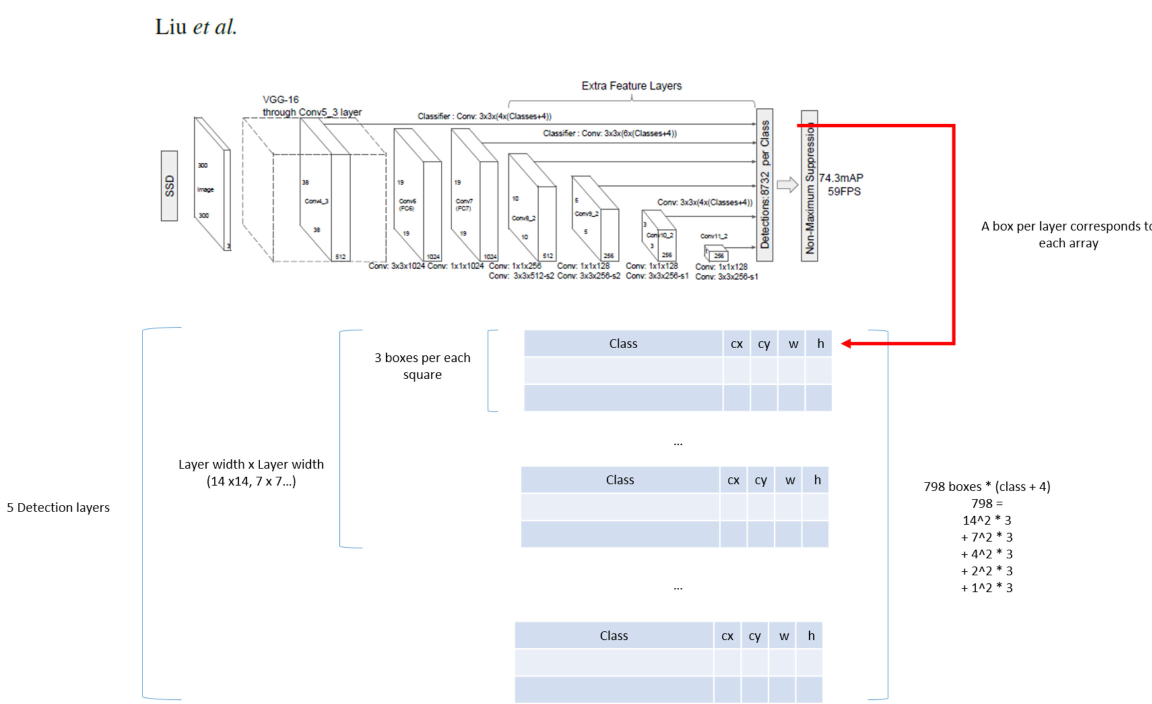
SSDInputEncodingGenerator

cifar10 데이터를 batch 단위로 입력받아 아래의 작업을 통해 SSD를 위한 데이터를 생성한다.

1. 32x32 이미지 데이터를 입력받은 뒤, 64 x 64로 확대하고, 224x224 검은 이미지에 랜덤하게 배치하여 트레이닝용 이미지로 한다. 이때 배치한 좌표의 위치를 Ground Truth의 좌표로 정의하고 다음 단계에서 적절한 Label 포맷으로 변형한다.



2. Detection용 Label을 생성한다. SSD에는 총 6개의 Detection 레이어가 있고, 각 레이어마다 격자를 나누고 4개/6개의 서로 다른 모양을 갖는 Anchor box를 배치한다. 각 Anchor box는 4개의 좌표값과(cx, cy, w, h), 10개의 클래스별 확률로 정의된다. Default class는 Background로 정의하고 다음 단계에서 매칭된 Box들만 Class를 부여해준다.



3. 위에서 설명한 calc_iou, match_bipartite_greedy, match_multi 등의 함수를 이용하여 Anchor box중 Ground truth와 많이 겹치는 Box들을 찾는다. 이렇게 찾은 Anchor Box에 class를 지정해주고, 아래의 수식에 따라 loss 계산에 필요한 값들을 정의한다. d는 anchorbox, g는 ground truth를 의미하며 윗첨자들은 각 좌표값(cx, cy, w, h)에 해당된다.

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

In [12]:

```
class SSDInputEncodingGenerator(keras.utils.Sequence):
    def __init__(self,
                 img_height,
                 img_width,
                 layer_width,
                 n_classes,
                 num_boxes,
                 s_max,
                 s_min,
                 aspect_ratio,
                 pos_iou_threshold,
                 neg_iou_threshold,
                background_id,
                 images,
                 labels,
                 data_size,
                batch_size=32):
        #Consider Background class
        self.img_height = img_height
        self.img_width = img_width
        self.n_class_withbg = n_classes + 1  #Add background class
        self.num_boxes = num_boxes  #List of number of boxes in each classifier
        self.s_max = s_max # Largest scale of default box
        self.s_min = s_min # Smallest scale of default box
        self.aspect_ratio = aspect_ratio # List of aspect ratios
        self.layer_width = layer_width
        self.pos_iou_threshold = pos_iou_threshold
        self.neg_iou_threshold = neg_iou_threshold
        self.background_id = background_id
        self.batch_size=batch_size
        self.images = images
        self.labels = labels
        self.shuffle = False
        self.data_size = data_size

        self.xmin_random = np.random.randint(self.img_height - 64, size=[self.da
        self.ymin_random = np.random.randint(self.img_height - 64, size=[self.da

        self.on_epoch_end()

    def convert_image(self, image, label, indexes):
        """
        Convert classification data to object detection data
        Randomly locate image in the middle of black canvas
```

```
        Input
            x: Image, shape: (batch_size, image size, image size, #channels)
            y: label, shape: (batch_size, )
        output
            out_x: Image located in the random location of black canvas, shape:
            out_y: label and location of corners(xmin,ymin,xmax,ymax), shape: (b

        """
        orig_image_size = 64
        channels = image.shape[-1]

        #prepare black canvas
        canvas = np.zeros((self.batch_size, self.img_height, self.img_width, cha
        out_y = np.zeros((self.batch_size, 1, 5))

        xmin = self.xmin_random[indexes]
        ymin = self.ymin_random[indexes]
        xmax = xmin + orig_image_size
        ymax = ymin + orig_image_size


        resized = np.zeros((orig_image_size, orig_image_size, 3))
        for i in range(batch_size):
            resized = cv2.resize(image[i], dsize=(orig_image_size, orig_image_si
            canvas[i, xmin[i]:xmax[i], ymin[i]:ymax[i], :] = resized

        out_y[:, 0,0] = label[:,0]
        out_y[:, 0, -4:] = np.column_stack([xmin, ymin, xmax, ymax])

        return canvas, out_y

    def __getitem__(self, index):
        '''
        Generate one batch of data
        '''
        # Generate indexes of the batch
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

        # Generate data
        X, y = self.__data_generation(indexes)

        return X, y

    def on_epoch_end(self):
        '''
        Updates indexes after each epoch
        '''
        self.indexes = np.arange(self.data_size)
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    def __len__(self):
        'Denotes the number of batches per epoch'
        return int(np.floor(self.images.shape[0] / self.batch_size))

    def __data_generation(self, indexes):
        """
        Input: ground truth label,shape: (batch_size, #object per image, 1 + 4)
        Output: y_encoded, shape: (batch_size, sum of grid size of all classifie
        1. Create y_encoded template: (B, num_boxes, class + 4 + 4) 4 for gt coo
        2. For each ground truth, calculate iou of gt and anchor boxes
```

```python
        3. Find the highest matching anchor box per each gt and fill in y_encode
        4. Multi object matching
        5. Apply negative iou threshold
        6. Transform into Delta format
        """

        images, gt_label = self.convert_image(self.images[indexes], self.labels[

        # Make class vector to one hot format
        class_vector = np.eye(self.n_class_withbg)

        #layer_width=[14,7,4,2,1]
        for iw in range(len(layer_width)):
            # s_max = 0.9   s_min = 0.5
            s = s_min + (s_max - s_min) / (len(layer_width) - 1) * (len(layer_wi
            l = layer_width[iw]
            num_box = self.num_boxes[iw]        # num_boxes = [3,3,3,3,3]
            box_tensor = np.zeros((l * l * num_box, 4))
            ### 아래 실습하면서 완성
            for i in range(l):
                for j in range(l):
                    for box_idx in range(num_box):
                        box_tensor[(i * l + j) * num_box + box_idx, 0] = (0.5 +
                        box_tensor[(i * l + j) * num_box + box_idx, 1] = (0.5 +
                        # aspect_ratio = [1, 1/2, 2]
                        box_tensor[(i * l + j) * num_box + box_idx, 2] = math.sq
                        box_tensor[(i * l + j) * num_box + box_idx, 3] = 1 / mat
            ### 실습 끝

            box_tensor = convert_coord(box_tensor, type='centroid2corner')

            if iw == 0:
                boxes_tensor = box_tensor
            else:
                boxes_tensor = np.concatenate((boxes_tensor, box_tensor), axis =

            class_tensor = np.zeros((l * l * num_box , self.n_class_withbg))

            if iw == 0:
                classes_tensor = class_tensor
            else:
                classes_tensor = np.concatenate((classes_tensor, class_tensor),

        box_class_tensor= np.concatenate((classes_tensor, boxes_tensor, boxes_te
        y_encoded = np.tile(box_class_tensor, (self.batch_size, 1, 1))

        y_encoded[:, :, self.background_id] = 1 # All boxes are background boxes


        #Ground truth matching
        for i in range(self.batch_size):
            gt_one_label = gt_label[i]
            m = gt_one_label.shape[0]
            if gt_one_label.shape[0] == 0: continue # If there is no object, ski

            #Normalize ground truth
            gt_one_label[:,[-4,-2]] /= self.img_width
            gt_one_label[:,[-3,-1]] /= self.img_height

            #FInd the iou of ground truth and all anchor boxes
            similarities = calc_iou(gt_one_label[:,-4:], y_encoded[i, :, -4:])
```

```
            #Find the highest matching anchor box per each ground truth boxes
            matches = match_bipartite_greedy(similarities)

            #Convert ground truth class label to one hot encoding
            gt_class = np.array(gt_one_label[:,0], dtype=np.int)

            #Fill in y_encoded
            y_encoded[i, matches, :self.n_class_withbg] = class_vector[gt_class]
            y_encoded[i, matches, -8:-4] = gt_one_label[:,1:]

            #Set the matched anchor boxes to 0 to indicate they are matched befo
            similarities[:,matches] = 0


            #Multi object matching
            #Similar process to bipartite matching
            matches_anchor, matches_gt = match_multi(similarities, threshold=sel

            if len(matches_gt) > 0:

                y_encoded[i, matches_anchor, :self.n_class_withbg] = class_vecto
                y_encoded[i, matches_anchor, -8:-4] = gt_one_label[matches_gt,1:

                #Set the matched anchor boxes to 0 to indicate they are matched
                similarities[:,matches_anchor] = 0

            #All background boxes whose iou are greater than neg_iou_threshold
            # are set to neutral(neither background nor class)
            max_bg_similarities = np.amax(similarities, axis = 0)
            neutral_boxes = np.nonzero(max_bg_similarities >= self.neg_iou_thres
            y_encoded[i, neutral_boxes, self.background_id] = 0

        #Convert coordinate from corner 2 centroid
        y_encoded[:,:,:-4] = convert_coord(y_encoded[:,:,:-4], type='corner2cent
        #print(y_encoded[0,0])
        y_encoded = convert_coord(y_encoded, type='corner2centroid')
        #print(y_encoded[0,0])

        y_encoded[:,:,[-8, -7]] -= y_encoded[:,:,[-4, -3]] # (cx(gt) - cx(d_box)
        y_encoded[:,:,[-8, -7]] /= y_encoded[:,:,[-2, -1]] # (cx(gt) - cx(d_box)
        y_encoded[:,:,[-6, -5]] = np.log(y_encoded[:,:,[-6, -5]] / y_encoded[:,:

        return images, y_encoded
```

In [13]:

```
ssd_input_gen = SSDInputEncodingGenerator(IMG_SIZE,
                 IMG_SIZE,
                 layer_width=layer_width,
                 n_classes=n_classes,
                 num_boxes=num_boxes,
                 s_max=s_max,
                 s_min=s_min,
                 aspect_ratio=aspect_ratio,
                 pos_iou_threshold=pos_iou_threshold,
                 neg_iou_threshold=neg_iou_threshold,
                 background_id=10,
                 images=x_train,
                 labels=y_train,
```

```
                data_size=train_size,
                batch_size=batch_size)
```

데이터가 잘 생성되었는지 확인해 보자. Generator에서 이미지를 하나 생성한 후, 매칭된 Anchor box와 Ground Truth bounding box를 이미지상에 표시해 보자.

In [14]:

```
import matplotlib.patches as patches

def show(image, label, img_width, img_height):

    fig,ax = plt.subplots(1, figsize=(10,10))
    ax.imshow(image)
    gt_boxes = np.argwhere(label[:,10]==0)
    print(gt_boxes)
    for match in gt_boxes:
        anchor_box = label[match[0],-4:]
        gt_box = label[match[0],-8:-4]
        xmin = anchor_box[0] - anchor_box[2]/2
        ymin = anchor_box[1] - anchor_box[3]/2
        w = anchor_box[2]
        h = anchor_box[3]

        w_gt = math.exp(gt_box[2]) * anchor_box[2] * img_width
        h_gt = math.exp(gt_box[3]) * anchor_box[3] * img_width
        cx_gt = (gt_box[0] * anchor_box[2] + anchor_box[0]) * img_width
        cy_gt = (gt_box[1] * anchor_box[3] + anchor_box[1]) * img_width
        xmin_gt = (cx_gt - w_gt/2)
        ymin_gt = (cy_gt - h_gt/2)

        rect = patches.Rectangle((ymin_gt,xmin_gt),h_gt,w_gt,linewidth=1,edgecol
        ax.text(ymin_gt+1, xmin_gt+5, 'Ground truth box', color='g')
        ax.add_patch(rect)

        xmin *= img_width
        ymin *= img_height
        w *= img_width
        h *= img_height

        rect = patches.Rectangle((ymin,xmin),h,w,linewidth=1,edgecolor='b',facec
        ax.text(ymin+1, xmin+5, 'Matched anchor box: {}'.format(match[0]), color
        ax.add_patch(rect)
    plt.show()
```

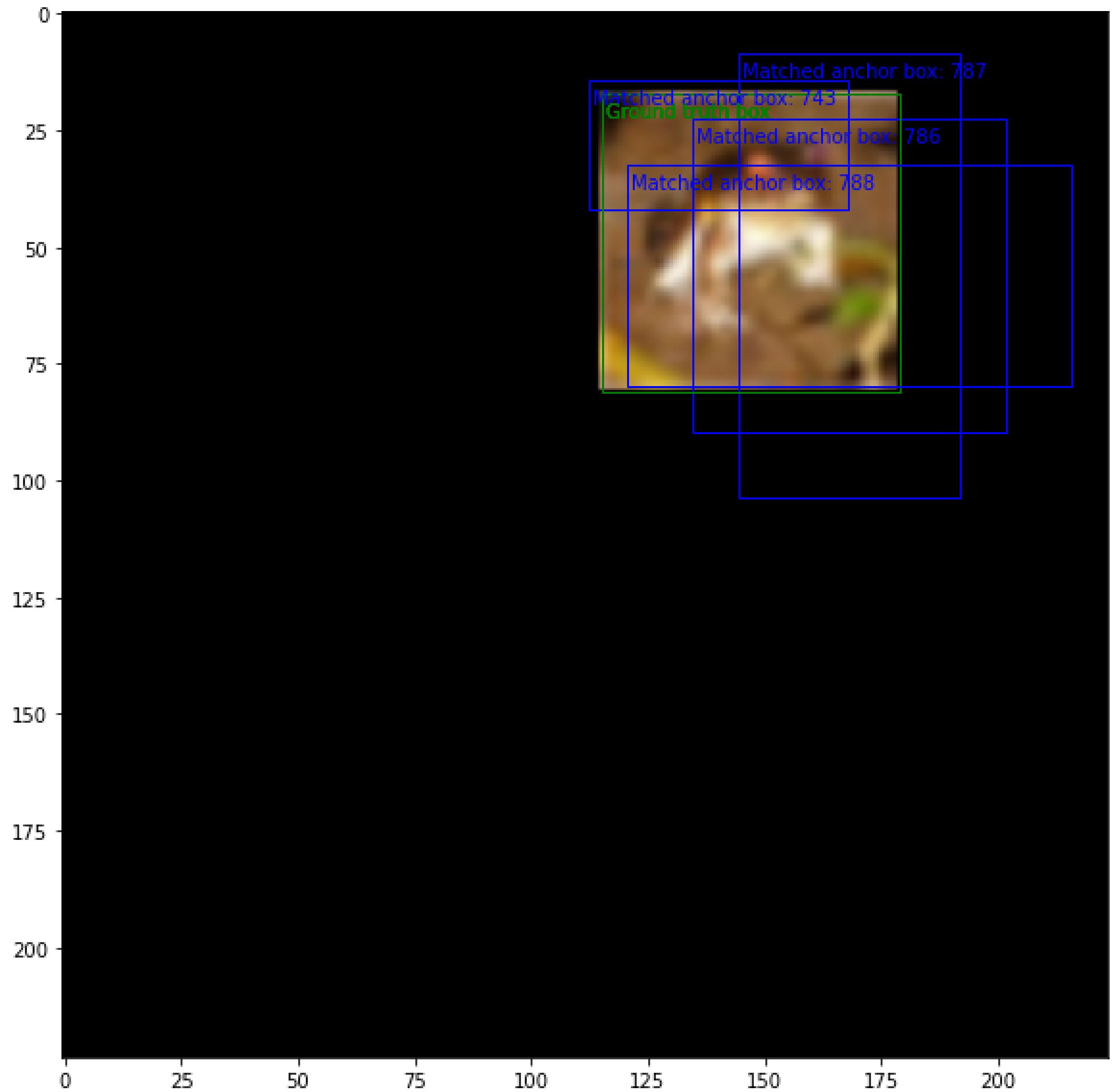
In [15]:

```
image, label = next(iter(ssd_input_gen))
```

In [16]:

```
show(image[0], label[0], IMG_SIZE, IMG_SIZE)
```

```
[[743]
 [786]
 [787]
 [788]]
```



Construct a model

이제 모델을 구성할 차례이다. 지난 실습때 구성한 MobiletNet V2를 기본으로 하여 Detection Layer들을 추가해 주는 방식으로 진행한다. 아래 두 함수(_make_divisible, _inverted_res_block)는 MobileNet V2와 동일한 Helper Function이다.

In [17]:

```
def _make_divisible(v, divisor, min_value=None):
    if min_value is None:
        min_value = divisor
    new_v = max(min_value, int(v + divisor / 2) // divisor * divisor) # 더 가까운
    # Make sure that round down does not go down by more than 10%.
    if new_v < 0.9 * v:
        new_v += divisor
    return new_v
```

In [18]:

```
def _inverted_res_block(inputs, expansion, stride, alpha, filters, block_id):
    #Get the channel axis and the input channel size
    channel_axis = 1 if backend.image_data_format() == 'channels_first' else -1
    in_channels = backend.int_shape(inputs)[channel_axis]
```

```
    pointwise_conv_filters = int(alpha * filters)
    pointwise_filters = _make_divisible(pointwise_conv_filters, 8) # Make sure t

    #Set the prefix
    prefix = 'block_{}_'.format(block_id)

    x = inputs

    #Expansion block
    if block_id: # No expansion for block 0
        x = layers.Conv2D(filters = expansion * in_channels, kernel_size = 1, st
                          use_bias=False, activation=None, kernel_regularizer=re
                          name=prefix + 'expand')(x)
        x = layers.BatchNormalization(axis=channel_axis, momentum=0.999, epsilon
                                      name=prefix + 'expand_BN')(x)
        x = layers.ReLU(6, name=prefix + 'expand_relu')(x)
    else:
        prefix = 'expanded_conv_'


    #Depthwise convolution
    #if stride == 2:
        #Adjust zero paddings for strides, when input hieght and width are odd a
        #x = layers.ZeroPadding2D(padding=correct_pad(x, 3),
        #                          name=prefix + 'pad')(x)

    x = layers.DepthwiseConv2D(kernel_size = 3, strides = stride,
                               #padding='same' if stride == 1 else 'valid',
                               padding='same',
                               use_bias=False, activation=None, kernel_regulariz
                               name=prefix + 'depthwise')(x)
    x = layers.BatchNormalization(axis=channel_axis, momentum=0.999, epsilon=0.0
                                  name=prefix + 'depthwise_BN')(x)
    x = layers.ReLU(6, name=prefix + 'relu')(x)

    #Pointwise convolution(Bottleneck)
    x = layers.Conv2D(filters = pointwise_filters, kernel_size = 1, strides = 1,
                      use_bias=False, activation=None, kernel_regularizer=regula
                      name=prefix + 'project')(x)
    x = layers.BatchNormalization(axis=channel_axis, momentum=0.999, epsilon=0.0
                                  name=prefix + 'project_BN')(x)

    #Inverted residual only when valid(Input size = output_size)
    if in_channels == pointwise_filters and stride == 1:
        return layers.add([inputs, x])
    return x
```

추후 좌표 복원의 편의를 위하여 각 Predicted box별로 Default anchor box의 좌표들을 붙여주는데, 아래는 이를 생성하기 위한 코드이다. Default anchor box의 좌표 생성 과정은 Training label 생성 과정과 동일하다.

In [19]:

```
from tensorflow.keras.layers import Layer, InputSpec

class AnchorBoxes(Layer):
    def __init__(self, layer_width, n_class_withbg, num_boxes,
                 s_max, s_min, aspect_ratio, index, **kwargs):
        self.layer_width = layer_width
        self.n_class_withbg = n_class_withbg
        self.num_boxes = num_boxes
        self.s_max = s_max
```

```
        self.s_min = s_min
        self.aspect_ratio = aspect_ratio
        self.index = index
        super(AnchorBoxes, self).__init__(**kwargs)

    def build(self, input_shape):
        self.input_spec = [InputSpec(shape=input_shape)]
        super(AnchorBoxes, self).build(input_shape)

    def compute_output_shape(self, input_shape):
        if K.image_dim_ordering() == 'tf':
            batch_size, feature_map_height, feature_map_width, feature_map_chann

        return (batch_size, feature_map_height*feature_map_width*self.n_boxes, 4

    def get_config(self):
        config = {
            'layer_width': list(self.layer_width),
            'n_class_withbg': self.n_class_withbg,
            'num_boxes': self.num_boxes,
            's_max': self.s_max,
            's_min': self.s_min,
            'aspect_ratio': list(self.aspect_ratio)
        }
        base_config = super(AnchorBoxes, self).get_config()
        return dict(list(base_config.items()) + list(config.items()))

    def call(self, x, mask=None):
        s = self.s_min + (self.s_max - self.s_min) / (len(self.layer_width) - 1)
        l = self.layer_width[self.index]
        num_box = self.num_boxes[self.index]
        box_tensor = np.zeros((l * l * num_box, 4))
        for i in range(self.layer_width[self.index]):
            for j in range(self.layer_width[self.index]):
                for box_idx in range(num_box):
                    box_tensor[(i * l + j) * num_box + box_idx, 0]  = (0.5 + i)
                    box_tensor[(i * l + j) * num_box + box_idx, 1]  = (0.5 + j)
                    box_tensor[(i * l + j) * num_box + box_idx, 2]  = math.sqrt(
                    box_tensor[(i * l + j) * num_box + box_idx, 3]  = 1 / math.s


        box_tensor = np.expand_dims(box_tensor, axis = 0)
        return tf.tile(tf.constant(box_tensor, dtype=tf.float32), (tf.shape(x)[0
```

MobileNet V2 모델을 변형하여 MobileNetV2SSD로 만들어 주자. 4번째 Bottleneck block을 통과한 뒤 첫 번째 Detection Layer(가장 촘촘한 격자)가 연결되도록 해주고, 2개의 Convolution layer를 통과한 뒤 두번째 Detection Layer가 연결되도록 해주자. 3~5번째 Detection layer는 2번째 Detection Layer와 동일한 방식으로 차례로 연결되도록 구성한다.

Note:

- Classification과 Localization을 위한 Layer를 따로 생성한 뒤, 나중에 Concatenate를 통해 합쳐주는 방식으로 구성한다. 하나의 layer로 생성하게되면, Classification에 Softmax를 적용하기가 어렵다.
- 위에 정의된 Anchor Box 함수를 이용하여 Default Anchor Box의 좌표를 상수로 입력해 둔다. 나중에 좌표 복원시에 용이하게 사용할 수 있다.

In [20]:

```
def MobileNetV2SSD(input_shape,
```

```
                 n_classes,
                 layer_width,
                 num_boxes,
                 alpha=1.0):

    n_class_withbg = n_classes + 1 # Add background class

    inputs = layers.Input(shape=input_shape)

    first_block_filters = _make_divisible(32 * alpha, 8)
    # first conv layer: 224x224x3 -> 112x112x32
    x = layers.Conv2D(first_block_filters, kernel_size=3, strides=(2, 2), paddin
                      bias_initializer='zeros',  kernel_regularizer=regularizers
                      name='Conv1')(inputs)

    x = layers.BatchNormalization(
      axis=-1, epsilon=1e-3, momentum=0.999, name='bn_Conv1')(x)

    x = layers.ReLU(6., name='Conv1_relu')(x)


    # inverted residual blocks
    # 1st bottleneck block: 112x112x32 -> 112x112x16
    x = _inverted_res_block(
      x, filters=16, alpha=alpha, stride=1, expansion=1, block_id=0)

    # 2nd bottleneck block: 112x112x16 -> 56x56x24
    x = _inverted_res_block(
      x, filters=24, alpha=alpha, stride=2, expansion=6, block_id=1)
    x = _inverted_res_block(
      x, filters=24, alpha=alpha, stride=1, expansion=6, block_id=2)

    # 3rd bottleneck block: 56x56x24 -> 28x28x32
    x = _inverted_res_block(
      x, filters=32, alpha=alpha, stride=2, expansion=6, block_id=3)
    x = _inverted_res_block(
      x, filters=32, alpha=alpha, stride=1, expansion=6, block_id=4)
    x = _inverted_res_block(
      x, filters=32, alpha=alpha, stride=1, expansion=6, block_id=5)

    # 4th bottleneck block: 28x28x32 -> 14x14x64
    x = _inverted_res_block(
      x, filters=64, alpha=alpha, stride=2, expansion=6, block_id=6)
    x = _inverted_res_block(
      x, filters=64, alpha=alpha, stride=1, expansion=6, block_id=7)
    x = _inverted_res_block(
      x, filters=64, alpha=alpha, stride=1, expansion=6, block_id=8)
    x = _inverted_res_block(
      x, filters=64, alpha=alpha, stride=1, expansion=6, block_id=9)

    classifier_1_conf = layers.Conv2D(num_boxes[0] * n_class_withbg, kernel_size
    classifier_1_loc = layers.Conv2D(num_boxes[0] * 4, kernel_size = 3, padding=


    x = layers.Conv2D(256, kernel_size=1, padding='same', use_bias=False, activa
    x = layers.Conv2D(512, kernel_size=3, strides=2, padding='same', use_bias=Fa

    classifier_2_conf = layers.Conv2D(num_boxes[1] * n_class_withbg, kernel_size
    classifier_2_loc = layers.Conv2D(num_boxes[1] * 4, kernel_size = 3, padding=

    x = layers.Conv2D(128, kernel_size=1, padding='same', use_bias=False, activa
```

```python
    x = layers.Conv2D(256, kernel_size=3, strides=2, padding='same', use_bias=Fa

    classifier_3_conf = layers.Conv2D(num_boxes[2] * n_class_withbg, kernel_size
    classifier_3_loc = layers.Conv2D(num_boxes[2] * 4, kernel_size = 3, padding=

    x = layers.Conv2D(128, kernel_size=1, padding='same', use_bias=False, activa
    x = layers.Conv2D(256, kernel_size=3, strides=2, padding='same', use_bias=Fa

    classifier_4_conf = layers.Conv2D(num_boxes[3] * n_class_withbg, kernel_size
    classifier_4_loc = layers.Conv2D(num_boxes[3] * 4, kernel_size = 3, padding=

    x = layers.Conv2D(128, kernel_size=1, padding='same', use_bias=False, activa
    x = layers.Conv2D(256, kernel_size=3, strides=2, padding='same', use_bias=Fa

    classifier_5_conf = layers.Conv2D(num_boxes[4] * n_class_withbg, kernel_size
    classifier_5_loc = layers.Conv2D(num_boxes[4] * 4, kernel_size = 3, padding=


    ### 아래 실습하면서 완성
    #Classification tensors
    classifier_1_conf = layers.Reshape((layer_width[0] * layer_width[0] * num_bo
    classifier_2_conf = layers.Reshape((layer_width[1] * layer_width[1] * num_bo
    classifier_3_conf = layers.Reshape((layer_width[2]*layer_width[2]*num_boxes[
    classifier_4_conf = layers.Reshape((layer_width[3]*layer_width[3]*num_boxes[
    classifier_5_conf = layers.Reshape((layer_width[4]*layer_width[4]*num_boxes[

    conf_layers = layers.concatenate([classifier_1_conf, classifier_2_conf, clas


    #Apply softmax
    conf_layers_softmax = layers.Activation('softmax')(conf_layers)

    #Localization tensors
    classifier_1_loc = layers.Reshape((layer_width[0] * layer_width[0] * num_box
    classifier_2_loc = layers.Reshape((layer_width[1] * layer_width[1] * num_box
    classifier_3_loc = layers.Reshape((layer_width[2]*layer_width[2]*num_boxes[2
    classifier_4_loc = layers.Reshape((layer_width[3]*layer_width[3]*num_boxes[3
    classifier_5_loc = layers.Reshape((layer_width[4]*layer_width[4]*num_boxes[4

    loc_layers = layers.concatenate([classifier_1_loc, classifier_2_loc, classif

    #Default anchor box tensors, They are constant and NOT trained !!
    #def __init__(self, layer_width, n_class_withbg, num_boxes, s_max, s_min, as
    dbox_1 = AnchorBoxes(layer_width, n_class_withbg, num_boxes, s_max, s_min, a
    dbox_2 = AnchorBoxes(layer_width, n_class_withbg, num_boxes, s_max, s_min, a
    dbox_3 = AnchorBoxes(layer_width, n_class_withbg, num_boxes, s_max, s_min, a
    dbox_4 = AnchorBoxes(layer_width, n_class_withbg, num_boxes, s_max, s_min, a
    dbox_5 = AnchorBoxes(layer_width, n_class_withbg, num_boxes, s_max, s_min, a

    dbox_layers = layers.concatenate([dbox_1, dbox_2, dbox_3, dbox_4, dbox_5], a

    ### 실습 끝


    #Concatenate Classification tensor, Localization tensor and Default anchor b
    detections = layers.concatenate([conf_layers_softmax, loc_layers, dbox_layer

    outputs = detections

    return Model(inputs=inputs, outputs=outputs)
```

In [21]:

```python
model = MobileNetV2SSD((IMG_SIZE, IMG_SIZE, 3), n_classes, layer_width, num_boxe
```

In [22]:

```python
model.summary()
```

```
Model: "model"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
________________________________________________________________________________
__________________
Conv1 (Conv2D)                  (None, 112, 112, 32) 864         input_1[0][0]
________________________________________________________________________________
__________________
bn_Conv1 (BatchNormalization)   (None, 112, 112, 32) 128         Conv1[0][0]
________________________________________________________________________________
__________________
Conv1_relu (ReLU)               (None, 112, 112, 32) 0           bn_Conv1[0][0]
________________________________________________________________________________
__________________
expanded_conv_depthwise (Depthw (None, 112, 112, 32) 288         Conv1_relu[0]
[0]
________________________________________________________________________________
__________________
expanded_conv_depthwise_BN (Bat (None, 112, 112, 32) 128         expanded_conv_d
epthwise[0][0]
________________________________________________________________________________
__________________
expanded_conv_relu (ReLU)       (None, 112, 112, 32) 0           expanded_conv_d
epthwise_BN[0][0]
________________________________________________________________________________
__________________
expanded_conv_project (Conv2D)  (None, 112, 112, 16) 512         expanded_conv_r
elu[0][0]
________________________________________________________________________________
__________________
expanded_conv_project_BN (Batch (None, 112, 112, 16) 64          expanded_conv_p
roject[0][0]
________________________________________________________________________________
__________________
block_1_expand (Conv2D)         (None, 112, 112, 96) 1536        expanded_conv_p
roject_BN[0][0]
________________________________________________________________________________
__________________
block_1_expand_BN (BatchNormali (None, 112, 112, 96) 384         block_1_expand
[0][0]
________________________________________________________________________________
__________________
block_1_expand_relu (ReLU)      (None, 112, 112, 96) 0           block_1_expand_
BN[0][0]
________________________________________________________________________________
__________________
block_1_depthwise (DepthwiseCon (None, 56, 56, 96)   864         block_1_expand_
relu[0][0]
________________________________________________________________________________
__________________
block_1_depthwise_BN (BatchNorm (None, 56, 56, 96)   384         block_1_depthwi
se[0][0]
________________________________________________________________________________
```

```
__________________
block_1_relu (ReLU)             (None, 56, 56, 96)   0           block_1_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_1_project (Conv2D)        (None, 56, 56, 24)   2304        block_1_relu[0]
[0]
________________________________________________________________________________
__________________
block_1_project_BN (BatchNormal (None, 56, 56, 24)   96          block_1_project
[0][0]
________________________________________________________________________________
__________________
block_2_expand (Conv2D)         (None, 56, 56, 144)  3456        block_1_project
_BN[0][0]
________________________________________________________________________________
__________________
block_2_expand_BN (BatchNormali (None, 56, 56, 144)  576         block_2_expand
[0][0]
________________________________________________________________________________
__________________
block_2_expand_relu (ReLU)      (None, 56, 56, 144)  0           block_2_expand_
BN[0][0]
________________________________________________________________________________
__________________
block_2_depthwise (DepthwiseCon (None, 56, 56, 144)  1296        block_2_expand_
relu[0][0]
________________________________________________________________________________
__________________
block_2_depthwise_BN (BatchNorm (None, 56, 56, 144)  576         block_2_depthwi
se[0][0]
________________________________________________________________________________
__________________
block_2_relu (ReLU)             (None, 56, 56, 144)  0           block_2_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_2_project (Conv2D)        (None, 56, 56, 24)   3456        block_2_relu[0]
[0]
________________________________________________________________________________
__________________
block_2_project_BN (BatchNormal (None, 56, 56, 24)   96          block_2_project
[0][0]
________________________________________________________________________________
__________________
add (Add)                       (None, 56, 56, 24)   0           block_1_project
_BN[0][0]
                                                                 block_2_project
_BN[0][0]
________________________________________________________________________________
__________________
block_3_expand (Conv2D)         (None, 56, 56, 144)  3456        add[0][0]
________________________________________________________________________________
__________________
block_3_expand_BN (BatchNormali (None, 56, 56, 144)  576         block_3_expand
[0][0]
________________________________________________________________________________
__________________
block_3_expand_relu (ReLU)      (None, 56, 56, 144)  0           block_3_expand_
BN[0][0]
________________________________________________________________________________
__________________
block_3_depthwise (DepthwiseCon (None, 28, 28, 144)  1296        block_3_expand_
relu[0][0]
________________________________________________________________________________
```

```
__________________
block_3_depthwise_BN (BatchNorm (None, 28, 28, 144)  576         block_3_depthwi
se[0][0]
________________________________________________________________________________
__________________
block_3_relu (ReLU)             (None, 28, 28, 144)  0           block_3_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_3_project (Conv2D)        (None, 28, 28, 32)   4608        block_3_relu[0]
[0]
________________________________________________________________________________
__________________
block_3_project_BN (BatchNormal (None, 28, 28, 32)   128         block_3_project
[0][0]
________________________________________________________________________________
__________________
block_4_expand (Conv2D)         (None, 28, 28, 192)  6144        block_3_project
_BN[0][0]
________________________________________________________________________________
__________________
block_4_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_4_expand
[0][0]
________________________________________________________________________________
__________________
block_4_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_4_expand_
BN[0][0]
________________________________________________________________________________
__________________
block_4_depthwise (DepthwiseCon (None, 28, 28, 192)  1728        block_4_expand_
relu[0][0]
________________________________________________________________________________
__________________
block_4_depthwise_BN (BatchNorm (None, 28, 28, 192)  768         block_4_depthwi
se[0][0]
________________________________________________________________________________
__________________
block_4_relu (ReLU)             (None, 28, 28, 192)  0           block_4_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_4_project (Conv2D)        (None, 28, 28, 32)   6144        block_4_relu[0]
[0]
________________________________________________________________________________
__________________
block_4_project_BN (BatchNormal (None, 28, 28, 32)   128         block_4_project
[0][0]
________________________________________________________________________________
__________________
add_1 (Add)                     (None, 28, 28, 32)   0           block_3_project
_BN[0][0]
                                                                 block_4_project
_BN[0][0]
________________________________________________________________________________
__________________
block_5_expand (Conv2D)         (None, 28, 28, 192)  6144        add_1[0][0]
________________________________________________________________________________
__________________
block_5_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_5_expand
[0][0]
________________________________________________________________________________
__________________
block_5_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_5_expand_
BN[0][0]
________________________________________________________________________________
```

```
__________________
block_5_depthwise (DepthwiseCon (None, 28, 28, 192)  1728        block_5_expand_
relu[0][0]
________________________________________________________________________________
__________________
block_5_depthwise_BN (BatchNorm (None, 28, 28, 192)  768         block_5_depthwi
se[0][0]
________________________________________________________________________________
__________________
block_5_relu (ReLU)             (None, 28, 28, 192)  0           block_5_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_5_project (Conv2D)        (None, 28, 28, 32)   6144        block_5_relu[0]
[0]
________________________________________________________________________________
__________________
block_5_project_BN (BatchNormal (None, 28, 28, 32)   128         block_5_project
[0][0]
________________________________________________________________________________
__________________
add_2 (Add)                     (None, 28, 28, 32)   0           add_1[0][0]
                                                                 block_5_project

_BN[0][0]
________________________________________________________________________________
__________________
block_6_expand (Conv2D)         (None, 28, 28, 192)  6144        add_2[0][0]
________________________________________________________________________________
__________________
block_6_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_6_expand
[0][0]
________________________________________________________________________________
__________________
block_6_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_6_expand_
BN[0][0]
________________________________________________________________________________
__________________
block_6_depthwise (DepthwiseCon (None, 14, 14, 192)  1728        block_6_expand_
relu[0][0]
________________________________________________________________________________
__________________
block_6_depthwise_BN (BatchNorm (None, 14, 14, 192)  768         block_6_depthwi
se[0][0]
________________________________________________________________________________
__________________
block_6_relu (ReLU)             (None, 14, 14, 192)  0           block_6_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_6_project (Conv2D)        (None, 14, 14, 64)   12288       block_6_relu[0]
[0]
________________________________________________________________________________
__________________
block_6_project_BN (BatchNormal (None, 14, 14, 64)   256         block_6_project
[0][0]
________________________________________________________________________________
__________________
block_7_expand (Conv2D)         (None, 14, 14, 384)  24576       block_6_project
_BN[0][0]
________________________________________________________________________________
__________________
block_7_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_7_expand
[0][0]
________________________________________________________________________________
__________________
```

```
block_7_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_7_expand_
BN[0][0]
________________________________________________________________________________
__________________
block_7_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_7_expand_
relu[0][0]
________________________________________________________________________________
__________________
block_7_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_7_depthwi
se[0][0]
________________________________________________________________________________
__________________
block_7_relu (ReLU)             (None, 14, 14, 384)  0           block_7_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_7_project (Conv2D)        (None, 14, 14, 64)   24576       block_7_relu[0]
[0]
________________________________________________________________________________
__________________
block_7_project_BN (BatchNormal (None, 14, 14, 64)   256         block_7_project
[0][0]
________________________________________________________________________________
__________________
add_3 (Add)                     (None, 14, 14, 64)   0           block_6_project
_BN[0][0]
                                                                 block_7_project
_BN[0][0]
________________________________________________________________________________
__________________
block_8_expand (Conv2D)         (None, 14, 14, 384)  24576       add_3[0][0]
________________________________________________________________________________
__________________
block_8_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_8_expand
[0][0]
________________________________________________________________________________
__________________
block_8_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_8_expand_
BN[0][0]
________________________________________________________________________________
__________________
block_8_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_8_expand_
relu[0][0]
________________________________________________________________________________
__________________
block_8_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_8_depthwi
se[0][0]
________________________________________________________________________________
__________________
block_8_relu (ReLU)             (None, 14, 14, 384)  0           block_8_depthwi
se_BN[0][0]
________________________________________________________________________________
__________________
block_8_project (Conv2D)        (None, 14, 14, 64)   24576       block_8_relu[0]
[0]
________________________________________________________________________________
__________________
block_8_project_BN (BatchNormal (None, 14, 14, 64)   256         block_8_project
[0][0]
________________________________________________________________________________
__________________
add_4 (Add)                     (None, 14, 14, 64)   0           add_3[0][0]
                                                                 block_8_project
_BN[0][0]
________________________________________________________________________________
```

```
__________________
block_9_expand (Conv2D)         (None, 14, 14, 384)  24576        add_4[0][0]
__________________________________________________________________________________
__________________
block_9_expand_BN (BatchNormali (None, 14, 14, 384)  1536         block_9_expand
[0][0]
__________________________________________________________________________________
__________________
block_9_expand_relu (ReLU)      (None, 14, 14, 384)  0            block_9_expand_
BN[0][0]
__________________________________________________________________________________
__________________
block_9_depthwise (DepthwiseCon (None, 14, 14, 384)  3456         block_9_expand_
relu[0][0]
__________________________________________________________________________________
__________________
block_9_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536         block_9_depthwi
se[0][0]
__________________________________________________________________________________
__________________
block_9_relu (ReLU)             (None, 14, 14, 384)  0            block_9_depthwi
se_BN[0][0]
__________________________________________________________________________________
__________________
block_9_project (Conv2D)        (None, 14, 14, 64)   24576        block_9_relu[0]
[0]
__________________________________________________________________________________
__________________
block_9_project_BN (BatchNormal (None, 14, 14, 64)   256          block_9_project
[0][0]
__________________________________________________________________________________
__________________
add_5 (Add)                     (None, 14, 14, 64)   0            add_4[0][0]
                                                                  block_9_project

_BN[0][0]
__________________________________________________________________________________
__________________
conv2d (Conv2D)                 (None, 14, 14, 256)  16384        add_5[0][0]
__________________________________________________________________________________
__________________
conv2d_1 (Conv2D)               (None, 7, 7, 512)    1179648      conv2d[0][0]
__________________________________________________________________________________
__________________
conv2d_2 (Conv2D)               (None, 7, 7, 128)    65536        conv2d_1[0][0]
__________________________________________________________________________________
__________________
conv2d_3 (Conv2D)               (None, 4, 4, 256)    294912       conv2d_2[0][0]
__________________________________________________________________________________
__________________
conv2d_4 (Conv2D)               (None, 4, 4, 128)    32768        conv2d_3[0][0]
__________________________________________________________________________________
__________________
conv2d_5 (Conv2D)               (None, 2, 2, 256)    294912       conv2d_4[0][0]
__________________________________________________________________________________
__________________
conv2d_6 (Conv2D)               (None, 2, 2, 128)    32768        conv2d_5[0][0]
__________________________________________________________________________________
__________________
conv2d_7 (Conv2D)               (None, 1, 1, 256)    294912       conv2d_6[0][0]
__________________________________________________________________________________
__________________
classifier_1_conf (Conv2D)      (None, 14, 14, 33)   19008        add_5[0][0]
__________________________________________________________________________________
__________________
classifier_2_conf (Conv2D)      (None, 7, 7, 33)     152064       conv2d_1[0][0]
```

```
____________________________________________________________________________________________________
____________________
classifier_3_conf (Conv2D)      (None, 4, 4, 33)     76032        conv2d_3[0][0]
____________________________________________________________________________________________________
____________________
classifier_4_conf (Conv2D)      (None, 2, 2, 33)     76032        conv2d_5[0][0]
____________________________________________________________________________________________________
____________________
classifier_5_conf (Conv2D)      (None, 1, 1, 33)     76032        conv2d_7[0][0]
____________________________________________________________________________________________________
____________________
classifier_1_loc (Conv2D)       (None, 14, 14, 12)   6912         add_5[0][0]
____________________________________________________________________________________________________
____________________
classifier_2_loc (Conv2D)       (None, 7, 7, 12)     55296        conv2d_1[0][0]
____________________________________________________________________________________________________
____________________
classifier_3_loc (Conv2D)       (None, 4, 4, 12)     27648        conv2d_3[0][0]
____________________________________________________________________________________________________
____________________
classifier_4_loc (Conv2D)       (None, 2, 2, 12)     27648        conv2d_5[0][0]
____________________________________________________________________________________________________
____________________
classifier_5_loc (Conv2D)       (None, 1, 1, 12)     27648        conv2d_7[0][0]
____________________________________________________________________________________________________
____________________
reshape (Reshape)               (None, 588, 11)      0            classifier_1_co
nf[0][0]
____________________________________________________________________________________________________
____________________
reshape_1 (Reshape)             (None, 147, 11)      0            classifier_2_co
nf[0][0]
____________________________________________________________________________________________________
____________________
reshape_2 (Reshape)             (None, 48, 11)       0            classifier_3_co
nf[0][0]
____________________________________________________________________________________________________
____________________
reshape_3 (Reshape)             (None, 12, 11)       0            classifier_4_co
nf[0][0]
____________________________________________________________________________________________________
____________________
reshape_4 (Reshape)             (None, 3, 11)        0            classifier_5_co
nf[0][0]
____________________________________________________________________________________________________
____________________
reshape_5 (Reshape)             (None, 588, 4)       0            classifier_1_lo
c[0][0]
____________________________________________________________________________________________________
____________________
reshape_6 (Reshape)             (None, 147, 4)       0            classifier_2_lo
c[0][0]
____________________________________________________________________________________________________
____________________
reshape_7 (Reshape)             (None, 48, 4)        0            classifier_3_lo
c[0][0]
____________________________________________________________________________________________________
____________________
reshape_8 (Reshape)             (None, 12, 4)        0            classifier_4_lo
c[0][0]
____________________________________________________________________________________________________
____________________
reshape_9 (Reshape)             (None, 3, 4)         0            classifier_5_lo
c[0][0]
____________________________________________________________________________________________________
```

```
__________________
concatenate (Concatenate)       (None, 798, 11)      0            reshape[0][0]
                                                                  reshape_1[0][0]
                                                                  reshape_2[0][0]
                                                                  reshape_3[0][0]
                                                                  reshape_4[0][0]
________________________________________________________________________________
__________________
anchor_boxes (AnchorBoxes)      (None, 588, 4)       0            reshape_5[0][0]
________________________________________________________________________________
__________________
anchor_boxes_1 (AnchorBoxes)    (None, 147, 4)       0            reshape_6[0][0]
________________________________________________________________________________
__________________
anchor_boxes_2 (AnchorBoxes)    (None, 48, 4)        0            reshape_7[0][0]
________________________________________________________________________________
__________________
anchor_boxes_3 (AnchorBoxes)    (None, 12, 4)        0            reshape_8[0][0]
________________________________________________________________________________
__________________
anchor_boxes_4 (AnchorBoxes)    (None, 3, 4)         0            reshape_9[0][0]
________________________________________________________________________________
__________________
activation (Activation)         (None, 798, 11)      0            concatenate[0]
[0]
________________________________________________________________________________
__________________
concatenate_1 (Concatenate)     (None, 798, 4)       0            reshape_5[0][0]
                                                                  reshape_6[0][0]
                                                                  reshape_7[0][0]
                                                                  reshape_8[0][0]
                                                                  reshape_9[0][0]
________________________________________________________________________________
__________________
concatenate_2 (Concatenate)     (None, 798, 4)       0            anchor_boxes[0]
[0]
                                                                  anchor_boxes_1
[0][0]
                                                                  anchor_boxes_2
[0][0]
                                                                  anchor_boxes_3
[0][0]
                                                                  anchor_boxes_4
[0][0]
________________________________________________________________________________
__________________
concatenate_3 (Concatenate)     (None, 798, 19)      0            activation[0]
[0]
                                                                  concatenate_1
[0][0]
                                                                  concatenate_2
[0][0]
================================================================================
==================
Total params: 3,004,928
Trainable params: 2,995,520
Non-trainable params: 9,408
________________________________________________________________________________
__________________
```

Loss function

SSD Loss는 Tensorflow에서 기본적으로 제공하지 않으므로, 직접 생성해 주어야 한다.

- Localization loss: L2 Loss를 사용한다. Positive box에만 적용한다.(Ground truth와 매칭된 anchor box)
- Confidence loss: Cross entropy loss를 사용한다. 다만 Positive box와 Negative box를 구분하여 각각에 대한 Loss를 따로 구하고 합쳐준다.

In [23]:

```
class SSDLoss():
    def __init__(self, n_classes, background_id, neg_pos_ratio=3, n_neg_min=0, a
        self.neg_pos_ratio = neg_pos_ratio
        self.n_neg_min = 0
        self.alpha = alpha
        self.beta = beta
        self.background_id = background_id
        self.n_class_withbg = n_classes + 1

    def smoothL1Loss(self, loc_true, loc_pred):
        """
        y_true: ground truth localization tensor, shape: (batch_size, num_boxes,
        y_pred: predicted localization tensor, shape: (batch_size, num_boxes, 4)
        """
        diff = tf.abs(loc_pred - loc_true)
        l2_loss = diff ** 2

        return tf.reduce_sum(l2_loss, axis=-1)

    def log_loss(self, class_true, class_pred):
        #classification loss
        class_pred = tf.maximum(class_pred, 1e-15)
        log_loss = -tf.reduce_sum(class_true * tf.math.log(class_pred), axis=-1)
        return log_loss

    def compute_loss(self, y_true, y_pred):
        """
        y_true: (batch_size, # boxes, n_class_withbg + 4)
        y_pred: (batch_size, # boxes, n_class_withbg + 4)
        """

        #Get the size of tensor
        batch_size = tf.shape(y_true)[0]
        n_boxes = tf.shape(y_pred)[1]

        y_true = tf.cast(y_true, dtype=tf.float32)
        y_pred = tf.cast(y_pred, dtype=tf.float32)

        classification_loss = self.log_loss(y_true[:, :, :self.n_class_withbg],
        positives = tf.reduce_max(y_true[:, :, :(self.n_class_withbg-1)], axis=-

        # Loss for positive boxes
        pos_class_loss = tf.reduce_sum(self.log_loss(y_true[:, :, :self.n_class_

        negatives = y_true[:,:, self.background_id] # Class is background, (batc
        n_positives = tf.reduce_sum(positives) # number of positive boxes, singl

        # Loss for negative boxes
        neg_class_loss_all = classification_loss * negatives #(batch_size, n_box
        n_neg_losses = tf.math.count_nonzero(neg_class_loss_all, dtype=tf.int32)

        # Keep the number of negative boxes between n_neg_min and neg_pos_ratio
        n_negative_keep = tf.minimum(tf.maximum(self.neg_pos_ratio * tf.cast(n_p
```

```
        def f1():
            return tf.zeros([batch_size])

        def f2():
            #Resahpe neg_class_loss_all to 1d array
            neg_class_loss_all_1D = tf.reshape(neg_class_loss_all, [-1])

            # Find top 'n_negative_keep' boxes from neg_class_loss_all_1D
            values, indices = tf.nn.top_k(neg_class_loss_all_1D, k=n_negative_ke

            #Then create a mask for negative boxes: For selected box above, set
            negatives_keep = tf.scatter_nd(indices=tf.expand_dims(indices, axis=
                                           updates=tf.ones_like(indices, dtype=t
                                           shape=tf.shape(neg_class_loss_all_1D)
            negatives_keep = tf.cast(tf.reshape(negatives_keep, [batch_size, n_b

            #Finally compute negative loss
            neg_class_loss = tf.reduce_sum(classification_loss * negatives_keep,

            return neg_class_loss

        neg_class_loss = tf.cond(tf.equal(n_neg_losses, tf.constant(0)), f1, f2)

        class_loss = self.beta * pos_class_loss + neg_class_loss

        #localization loss
        loc_pred = y_pred[:,:,self.n_class_withbg:-4]
        loc_true = y_true[:,:,self.n_class_withbg:-4]
        loc_loss = self.smoothL1Loss(loc_true, loc_pred) # (batch_size, n_boxes)

        # Include only positive boxes in calculating localization loss
        loc_loss = tf.reduce_sum(positives * loc_loss, axis=-1) #(batch_size)

        #Combine localization and classification loss, divide by matched default
        total_loss = (class_loss + loc_loss * self.alpha) / tf.maximum(1.0, n_po

        # We divided by n_positives - # of all matched default boxes of "a batch
        # Since keras divides by the size of batch, it is double division
        # To adjust this, we multiply by batch_size
        total_loss = total_loss * tf.cast(batch_size, dtype=tf.float32)

        return total_loss
```

Model compile and Training

모델을 컴파일 하고 트레이닝을 시작하자.

- Optimizer: Adam을 사용하되, Learning schedule을 통해 Learning rate를 조정해 주자.
- Loss: 위에서 정의한 SSDLoss를 사용하자.

In [24]:

```
ssd_loss = SSDLoss(n_classes = n_classes, background_id=10, neg_pos_ratio=3, alp
model.compile(loss=ssd_loss.compute_loss,
              optimizer=tf.keras.optimizers.Adam())
              #optimizer=tf.keras.optimizers.SGD(lr=0.001, momentum=0.9, decay=0
```

In [25]:

```
#model.load_weights(os.path.join(checkpoint_dir, "ckpt_39"))
```

CallBack 함수를 지정하면 필요한 대로 트레이닝 옵션들을 추가할 수 있다.

In [26]:
```
#decay could be applied using Learning rate scheduler
def decay(epoch):
    return 0.001 * (0.98 **(epoch - 1))
```

In [27]:
```
callbacks = []
"""
#TensorBoard로 훈련 성과를 보고 싶은 경우
callbacks.append(TensorBoard(log_dir=log_dir, histogram_freq=1))
"""

#Checkpoint설정
checkpoint_dir = './training_checkpoints_SSD'
model_cp_path = os.path.join(checkpoint_dir, "ckpt_{epoch}")
callbacks.append(tf.keras.callbacks.ModelCheckpoint(model_cp_path, save_weights_


#Learning rate 스케쥴 설정
callbacks.append(LearningRateScheduler(decay))

#General logs on csv
callbacks.append(CSVLogger(model_csv_path))
```

In [28]:
```
history = model.fit(ssd_input_gen,
              epochs=1,
              verbose=1,
              callbacks=callbacks)
```

```
1562/1562 [==============================] - 663s 99ms/step - loss: 144.6845
```

Prediction and Evaluation

이제 트레이닝된 모델을 이용하여 Detection이 잘 되는지 확인해 보자. 먼저 Training image, label을 이용하여 Ground truth box와 비교해 볼 것이다. Training data generator로부터 나온 image batch를 model에 넣어 predict를 해보자.

In [29]:
```
y_pred = model.predict(image)
```

아래 함수들을 통해 y_pred로부터 탐지된 Bounding Box를 찾아낸다.

- greedy_nms: Non-max suppression을 수행한다. Confidence가 높게 예측된 Box들을 뽑아낸 후, 겹치는 비율(IoU)이 높은 Box들을 제거해 주는 함수이다.
- decode_detections: 예측된 Box들을 다시 원본 코드의 좌표로 복원해 주는 작업을 수행한다.

In [30]:
```
def _greedy_nms(predictions, iou_threshold=0.45):
    '''
    Non-maximum suppression.
    '''
    boxes_left = np.copy(predictions)
    maxima = [] # This is where we store the boxes that make it through the non-
    while boxes_left.shape[0] > 0: # While there are still boxes left to compare
        maximum_index = np.argmax(boxes_left[:,0]) # ...get the index of the nex
```

```
        maximum_box = np.copy(boxes_left[maximum_index]) # ...copy that box and.
        maxima.append(maximum_box) # ...append it to `maxima` because we'll defi
        boxes_left = np.delete(boxes_left, maximum_index, axis=0) # Now remove t
        if boxes_left.shape[0] == 0: break # If there are no boxes left after th
        similarities = calc_iou(boxes_left[:,1:],np.expand_dims(maximum_box[1:],
        boxes_left = boxes_left[(similarities <= iou_threshold)[:,0]] # ...so th
    return np.array(maxima)
```

In [31]:

```
def decode_detections(y_pred,
                      n_classes,
                      confidence_thresh=0.01,
                      iou_threshold=0.45,
                      top_k=200,
                      img_height=None,
                      img_width=None,
                      background_id=10):
    # 1: Convert the box coordinates from the predicted anchor box offsets to pr

    y_pred_decoded_raw = np.copy(y_pred[:,:,:-4]) # Slice out the classes and th

    # exp(ln(w(pred)/w(anchor)) / w_variance * w_variance) == w(pred) / w(anchor
    y_pred_decoded_raw[:,:,[-2,-1]] = np.exp(y_pred_decoded_raw[:,:,[-2,-1]])

    # (w(pred) / w(anchor)) * w(anchor) == w(pred), (h(pred) / h(anchor)) * h(an
    y_pred_decoded_raw[:,:,[-2,-1]] *= y_pred[:,:,[-2,-1]]

    # (delta_cx(pred) / w(anchor) / cx_variance) * cx_variance * w(anchor) == de
    y_pred_decoded_raw[:,:,[-4,-3]] *= y_pred[:,:,[-2,-1]]

    # delta_cx(pred) + cx(anchor) == cx(pred), delta_cy(pred) + cy(anchor) == cy
    y_pred_decoded_raw[:,:,[-4,-3]] += y_pred[:,:,[-4,-3]]
    y_pred_decoded_raw = convert_coord(y_pred_decoded_raw, type='centroid2corner

    # 2: If the model predicts normalized box coordinates and they are supposed
    y_pred_decoded_raw[:,:,[-4,-2]] *= img_width # Convert xmin, xmax back to ab
    y_pred_decoded_raw[:,:,[-3,-1]] *= img_height # Convert ymin, ymax back to a

    # 3: Apply confidence thresholding and non-maximum suppression per class

    #n_classes = y_pred_decoded_raw.shape[-1] - 4 # The number of classes is the

    y_pred_decoded = [] # Store the final predictions in this list
    for batch_item in y_pred_decoded_raw: # `batch_item` has shape `[n_boxes, n_
        pred = [] # Store the final predictions for this batch item here
        for class_id in range(n_classes): # For each class except the background
            if class_id == background_id: continue
            single_class = batch_item[:,[class_id, -4, -3, -2, -1]] # ...keep on
            which_box = np.argwhere(single_class[:,0] > confidence_thresh)
            threshold_met = single_class[single_class[:,0] > confidence_thresh]

            if threshold_met.shape[0] > 0: # If any boxes made the threshold...
                maxima = _greedy_nms(threshold_met, iou_threshold=iou_threshold)
                maxima_output = np.zeros((maxima.shape[0], maxima.shape[1] + 1))
                maxima_output[:,0] = class_id # Write the class ID to the first
                maxima_output[:,1:] = maxima # ...and write the maxima to the ot
                pred.append(maxima_output) # ...and append the maxima for this c
        # Once we're through with all classes, keep only the `top_k` maxima with
        if pred: # If there are any predictions left after confidence-thresholdi
            pred = np.concatenate(pred, axis=0)
```

```
            if top_k != 'all' and pred.shape[0] > top_k: # If we have more than
                top_k_indices = np.argpartition(pred[:,1], kth=pred.shape[0]-top
                pred = pred[top_k_indices] # ...and keep only those entries of `
        else:
            pred = np.array(pred) # Even if empty, `pred` must become a Numpy ar
        y_pred_decoded.append(pred) # ...and now that we're done, append the arr

    return y_pred_decoded
```

In [32]:

```
# Decode the prediction for one image in the image batch
image_no = 0
y_decoded = decode_detections(np.expand_dims(y_pred[image_no], axis=0),
                            n_classes=10,
                              confidence_thresh=0.01,
                              iou_threshold=0.45,
                              top_k=10,
                              img_height=IMG_SIZE,
                              img_width=IMG_SIZE,
                              background_id=10)
```

예측된 Bounding box를 Image 위에 시각화 해보자.

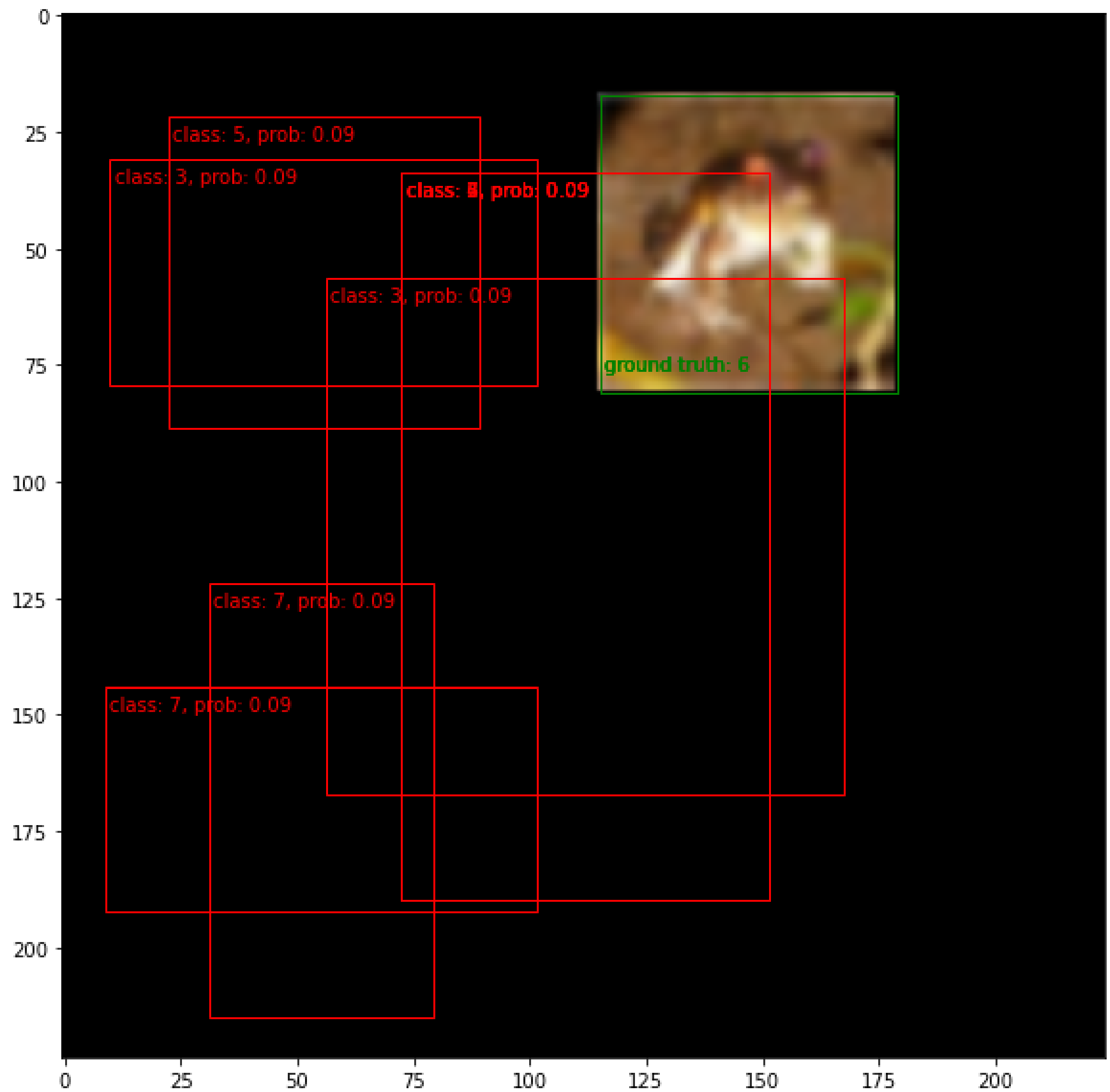
In [33]:

```
# Visualize the bounding box on the original image
import matplotlib.patches as patches

def show_prediction(image, label, prediction):

    fig,ax = plt.subplots(1, figsize=(10,10))
    ax.imshow(image)
    gt_boxes = np.argwhere(label[:,10]==0)
    for match in gt_boxes:
        anchor_box = label[match[0],-4:]
        gt_box = label[match[0], -8:-4]
        class_id = np.argwhere(label[match[0],:10]==1)

        w = math.exp(gt_box[2]) * anchor_box[2]
        h = math.exp(gt_box[3]) * anchor_box[3]
        cx = gt_box[0] * anchor_box[2] + anchor_box[0]
        cy = gt_box[1] * anchor_box[3] + anchor_box[1]

        xmin = (cx - w/2) * IMG_SIZE
        ymin = (cy - h/2) * IMG_SIZE
        w = w * IMG_SIZE
        h = h * IMG_SIZE

        rect = patches.Rectangle((ymin,xmin),h, w,linewidth=1,edgecolor='g',face
        ax.add_patch(rect)
        ax.text(ymin+1, xmin+w-5, 'ground truth: ' + str(class_id[0,0]), color='

    pred_boxes= np.argwhere(prediction[:,1] > 0)

    for pred in pred_boxes:
        box = prediction[pred[0],2:6]
        class_id = int(prediction[pred[0],0])
        prob = prediction[pred[0],1]
        xmin = min(max(box[0],0),224)
        ymin = min(max(box[1],0),224)
        w = min(max(box[2],0),224) - xmin
```

```
        h = min(max(box[3],0),224) - ymin
        rect = patches.Rectangle((ymin,xmin),h, w,linewidth=1,edgecolor='r',face
        ax.add_patch(rect)
        ax.text(ymin+1, xmin+5, 'class: {}, prob: {:.2f}'.format(class_id, prob)

    plt.show()
```

In [34]:

```
show_prediction(image[image_no],label[image_no],y_decoded[0])
```



이제 테스트 셋에 대해서도 예측을 해보고, Evaluation을 해 볼 차례이다. 먼저 SSDInputEncodingGenerator 클래스에 테스트셋(x_test, y_test)를 넣어 테스트용 Generator를 준비하도록 하자.

In [35]:

```
ssd_test_gen = SSDInputEncodingGenerator(IMG_SIZE,
                    IMG_SIZE,
                    layer_width=layer_width,
                    n_classes=n_classes,
                    num_boxes=num_boxes,
                    s_max=s_max,
                    s_min=s_min,
                    aspect_ratio=aspect_ratio,
                    pos_iou_threshold=pos_iou_threshold,
```

```
                  neg_iou_threshold=neg_iou_threshold,
                  background_id=10,
                  images=x_test,
                  labels=y_test,
                 data_size=test_size,
                 batch_size=32)
```

테스트 셋의 이미지에 대해서 예측이 잘 되는지 확인을 해보자.

In [36]:

```
test_image, test_label = next(iter(ssd_test_gen))
```

In [37]:

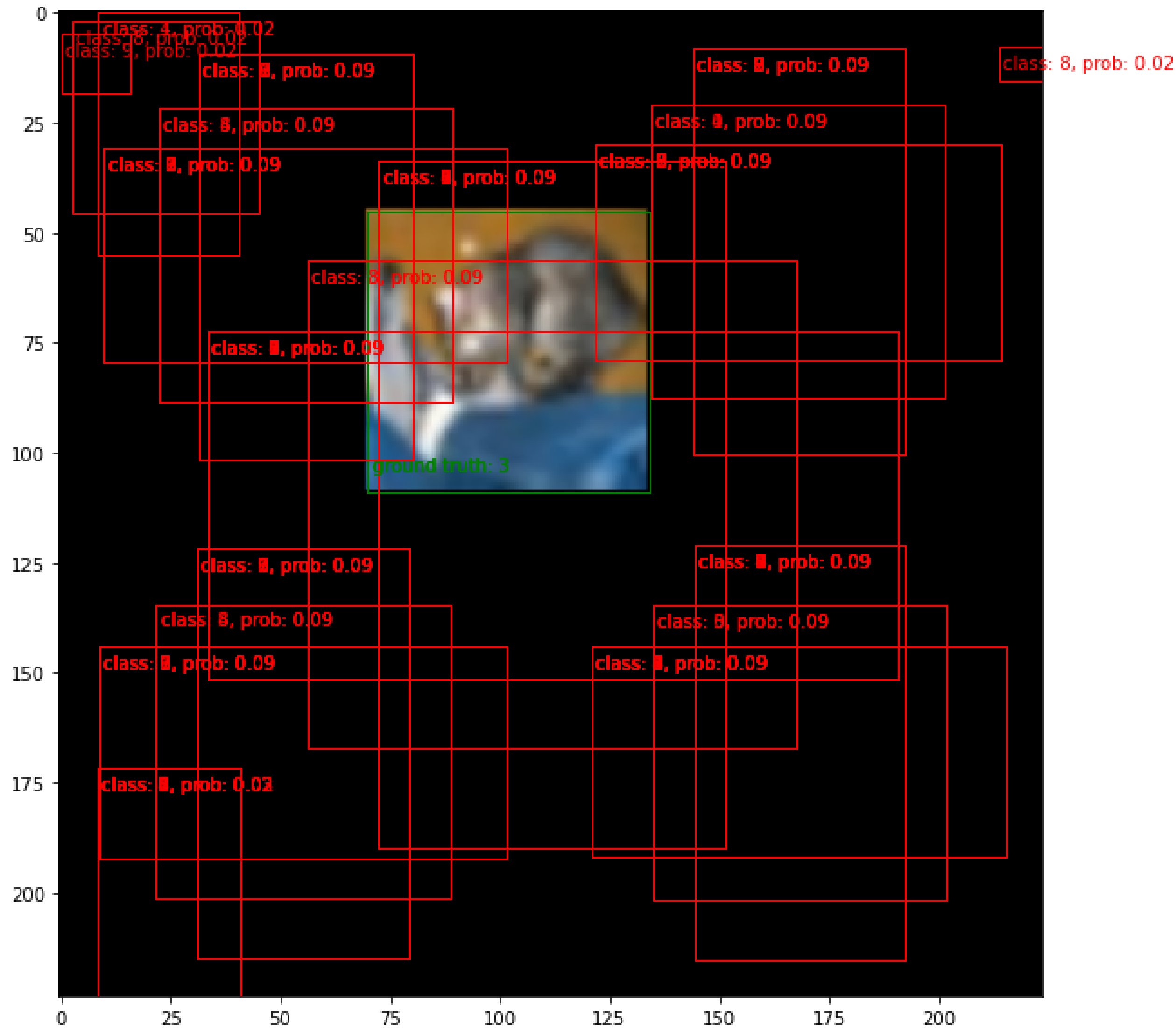
```
test_pred = model.predict(test_image)
```

In [38]:

```
image_no = 0
test_decoded = decode_detections(np.expand_dims(test_pred[image_no], axis=0),
                            n_classes=10,
                              confidence_thresh=0.01,
                              iou_threshold=0.45,
                              top_k=100,
                              img_height=IMG_SIZE,
                              img_width=IMG_SIZE,
                              background_id=10)
```

In [39]:

```
show_prediction(test_image[image_no],test_label[image_no],test_decoded[0])
```



Evaluation

이제 Detection 성능을 측정해 볼 것이다. Detection 성능은 보통 mAP(mean Average Precision)으로 측정한다. 여러 Challenge에 따라 조금씩 기준이 다른데 우리는 COCO mAP를 이용하여 성능을 측정해 보기로 한다.

- mAP에 관한설명 참고 사이트: https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/
- COCO challenge mAP 기준에 관한 설명: https://cocodataset.org/#detection-eval

COCO 기준의 mAP를 측정하기 위해서는 pycocotools 패키지를 사용하면 된다. 다만, Ground truth와 Detection 결과는 직접 json 형태로 만들어 주어야 한다. 아래 코드들을 통해 필요한 파일들을 준비할 수 있다.

In [40]:

```
# Function to dump prediction result in JSON format
import json

def dump_coco_json(dataset_size, batch_size, generator, model, out_file):

    # Put the results in this list.
    results = []
    id_cnt = 0
```

```python
    for batch_X, batch_label in generator:
        # Generate batch.
        #batch_X, batch_label = next(generator)
        # Predict.
        y_pred = model.predict(batch_X)

        # Decode.
        y_pred = decode_detections(y_pred,
                                    n_classes=10,
                                    confidence_thresh=0.01,
                                    iou_threshold=0.45,
                                    top_k=200,
                                    img_height=IMG_SIZE,
                                   img_width=IMG_SIZE,
                                   background_id=10)

        # Convert each predicted box into the results format.
        for k, batch_item in enumerate(y_pred):
            for box in batch_item:
                cat_id = box[0]
                # Round the box coordinates to reduce the JSON file size.
                xmin = float(round(box[2], 1))
                ymin = float(round(box[3], 1))
                xmax = float(round(box[4], 1))
                ymax = float(round(box[5], 1))
                width = xmax - xmin
                height = ymax - ymin
                bbox = [xmin, ymin, width, height]
                result = {}
                result['image_id'] = id_cnt
                result['category_id'] = cat_id
                result['score'] = float(round(box[1], 3))
                result['bbox'] = bbox
                results.append(result)
            id_cnt += 1
            if id_cnt == dataset_size:
                break


    with open(out_file, 'w') as f:
        json.dump(results, f)

    print("Prediction results saved in '{}'".format(out_file))
    return
```

In [41]:

```python
# Prediction to coco format
generator = ssd_test_gen
dataset_size = test_size
#dataset_size = 128
out_file='prediction_coco_format.json'
```

In [42]:

```python
dump_coco_json(dataset_size, batch_size, generator, model, out_file)
```

```
Prediction results saved in 'prediction_coco_format.json'
```

In [43]:

```python
#Get the coordinates of ground truth image from the generator
xmin_test = np.expand_dims(ssd_test_gen.xmin_random, axis=-1)
```

```
ymin_test = np.expand_dims(ssd_test_gen.ymin_random, axis=-1)
xmax_test = np.expand_dims(ssd_test_gen.xmin_random + 64, axis=-1)
ymax_test = np.expand_dims(ssd_test_gen.ymin_random + 64, axis=-1)

#Prepare ground truth boxes information(class_id, confidence(dummy), xmin, ymin,
gt_boxes = np.concatenate([y_test, np.ones([test_size,1]), xmin_test, ymin_test,
```

In [44]:

```
# Store gt information in coco format
images = []
results = []
categories = []

for i in range(dataset_size):
    im = {}
    im['id'] = i
    im['width'] = IMG_SIZE
    im['height'] = IMG_SIZE
    im['file_name'] = 'image.jpg'
    images.append(im)

for i in range(n_classes):
    cat = {}
    cat['id'] = i
    cat['name'] = class_names[i]
    cat['supercategory'] = cat['name']
    categories.append(cat)

id_cnt = 0

for box in gt_boxes:
    class_id = box[0]
    # Transform the consecutive class IDs back to the original COCO category IDs
    #cat_id = classes_to_cats[class_id]
    cat_id = class_id
    # Round the box coordinates to reduce the JSON file size.
    xmin = float(round(box[2], 1))
    ymin = float(round(box[3], 1))
    xmax = float(round(box[4], 1))
    ymax = float(round(box[5], 1))
    width = xmax - xmin
    height = ymax - ymin
    bbox = [xmin, ymin, width, height]
    result = {}
    result['id'] = id_cnt
    result['image_id'] = id_cnt
    result['category_id'] = cat_id
    result['bbox'] = bbox
    result['iscrowd'] = 0
    result['area'] = width * height
    results.append(result)
    id_cnt += 1

    if id_cnt == dataset_size:
        break

output_dict = {}
output_dict["images"] = images
output_dict["annotations"] = results
output_dict["categories"] = categories
```

```
out_file='gt_coco_format.json'

with open(out_file, 'w') as f:
    json.dump(output_dict, f)
```

필요한 파일들이 준비되었으며 pycocotools 패키지를 이용하여 Evaluation을 수행한다.

In [45]:
```
pip install pycocotools
```

```
Requirement already satisfied: pycocotools in /home/sungwookson/anaconda/lib/pyt
hon3.8/site-packages (2.0.2)
Requirement already satisfied: setuptools>=18.0 in /home/sungwookson/anaconda/li
b/python3.8/site-packages (from pycocotools) (52.0.0.post20210125)
Requirement already satisfied: cython>=0.27.3 in /home/sungwookson/anaconda/lib/
python3.8/site-packages (from pycocotools) (0.29.23)
Requirement already satisfied: matplotlib>=2.1.0 in /home/sungwookson/anaconda/l
ib/python3.8/site-packages (from pycocotools) (3.3.4)
Requirement already satisfied: cycler>=0.10 in /home/sungwookson/anaconda/lib/py
thon3.8/site-packages (from matplotlib>=2.1.0->pycocotools) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /home/sungwookson/anacond
a/lib/python3.8/site-packages (from matplotlib>=2.1.0->pycocotools) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in /home/sungwookson/anaconda/lib/p
ython3.8/site-packages (from matplotlib>=2.1.0->pycocotools) (8.2.0)
Requirement already satisfied: numpy>=1.15 in /home/sungwookson/anaconda/lib/pyt
hon3.8/site-packages (from matplotlib>=2.1.0->pycocotools) (1.19.5)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /hom
e/sungwookson/anaconda/lib/python3.8/site-packages (from matplotlib>=2.1.0->pyco
cotools) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /home/sungwookson/anaconda/l
ib/python3.8/site-packages (from matplotlib>=2.1.0->pycocotools) (1.3.1)
Requirement already satisfied: six in /home/sungwookson/anaconda/lib/python3.8/s
ite-packages (from cycler>=0.10->matplotlib>=2.1.0->pycocotools) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [46]:
```
from pycocotools.coco import COCO
from pycocotools.cocoeval import COCOeval
```

In [47]:
```
coco_gt = COCO('gt_coco_format.json')
```

```
loading annotations into memory...
Done (t=0.04s)
creating index...
index created!
```

In [48]:
```
coco_dt = coco_gt.loadRes('prediction_coco_format.json')
```

```
Loading and preparing results...
DONE (t=10.89s)
creating index...
index created!
```

In [49]:
```
image_ids = sorted(coco_gt.getImgIds())
```

In [50]:
```
cocoEval = COCOeval(cocoGt=coco_gt,
                    cocoDt=coco_dt,
                    iouType='bbox')
```

```
cocoEval.params.imgIds  = image_ids
cocoEval.evaluate()
cocoEval.accumulate()
cocoEval.summarize()
```

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=55.72s).
Accumulating evaluation results...
DONE (t=15.38s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.001
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.005
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.037
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.037
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.037
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
```

In [51]:

```
class evaluation_mAP(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        if (0 < epoch and epoch % 5 == 0) or epoch >= 10:
        #if epoch >= 0:
            print('\n')
            print('Start evalutaion')
            dump_coco_json(test_size, batch_size, ssd_test_gen, model, 'predicti
            coco_dt = coco_gt.loadRes('prediction_coco_format.json')
            image_ids = sorted(coco_gt.getImgIds())
            cocoEval = COCOeval(cocoGt=coco_gt,
                    cocoDt=coco_dt,
                    iouType='bbox')
            cocoEval.params.imgIds  = image_ids
            cocoEval.evaluate()
            cocoEval.accumulate()
            cocoEval.summarize()

callbacks.append(evaluation_mAP())
```

In [52]:

```
history = model.fit(ssd_input_gen,
              epochs=50,
              verbose=1,
              callbacks=callbacks)
```

```
Epoch 1/50
   6/1562 [..............................] - ETA: 2:33 - loss: 129.9876WARNING:t
ensorflow:Callback method `on_train_batch_end` is slow compared to the batch tim
e (batch time: 0.0349s vs `on_train_batch_end` time: 0.0634s). Check your callba
cks.
1562/1562 [==============================] - 154s 99ms/step - loss: 13.9815


Start evalutaion
Prediction results saved in 'prediction_coco_format.json'
Loading and preparing results...
DONE (t=0.13s)
creating index...
index created!
```

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=9.92s).
Accumulating evaluation results...
DONE (t=1.41s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.156
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.522
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.030
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.156
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.290
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.335
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.335
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.335
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
Epoch 26/50
1562/1562 [==============================] - 155s 99ms/step - loss: 13.0932


Start evalutaion
Prediction results saved in 'prediction_coco_format.json'
Loading and preparing results...
DONE (t=0.05s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=3.95s).
Accumulating evaluation results...
DONE (t=0.84s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.016
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.005
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.007
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.007
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.007
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
Epoch 27/50
1320/1562 [========================>.....] - ETA: 23s - loss: 12.4328
IOPub message rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_msg_rate_limit`.

Current values:
NotebookApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
NotebookApp.rate_limit_window=3.0 (secs)

1562/1562 [==============================] - 155s 99ms/step - loss: 7.8789


Start evalutaion
Prediction results saved in 'prediction_coco_format.json'
Loading and preparing results...
DONE (t=0.06s)
creating index...
index created!
```

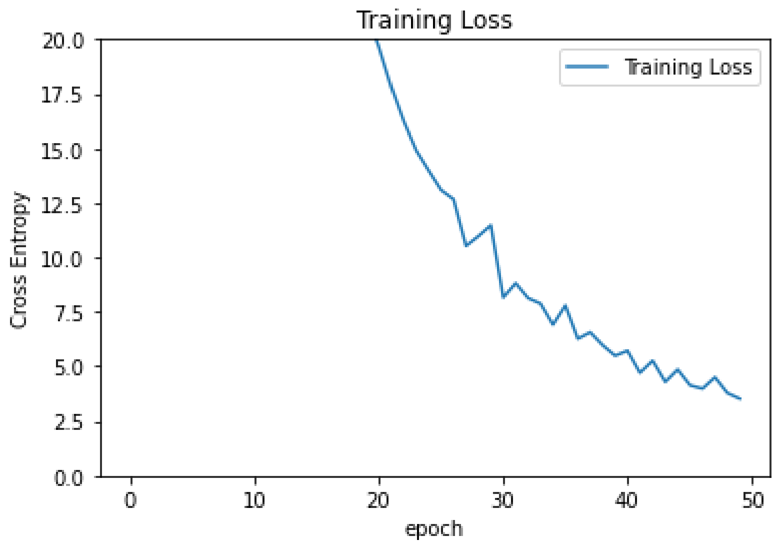
```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=4.33s).
Accumulating evaluation results...
DONE (t=0.97s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.004
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.016
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.004
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.006
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.009
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.009
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.009
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
Epoch 35/50
1562/1562 [==============================] - 156s 100ms/step - loss: 6.9093


Start evalutaion
Prediction results saved in 'prediction_coco_format.json'
Loading and preparing results...
DONE (t=0.02s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=3.13s).
Accumulating evaluation results...
DONE (t=0.66s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.011
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.002
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.004
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.005
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.005
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.005
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
Epoch 36/50
1079/1562 [===================>..........] - ETA: 48s - loss: 7.1554
IOPub message rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_msg_rate_limit`.

Current values:
NotebookApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
NotebookApp.rate_limit_window=3.0 (secs)

1562/1562 [==============================] - 156s 100ms/step - loss: 5.2454


Start evalutaion
Prediction results saved in 'prediction_coco_format.json'
Loading and preparing results...
DONE (t=0.08s)
creating index...
index created!
```

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=5.76s).
Accumulating evaluation results...
DONE (t=1.09s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.247
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.617
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.121
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.247
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.395
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.415
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.415
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.415
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
Epoch 44/50
1562/1562 [==============================] - 155s 99ms/step - loss: 4.2830


Start evalutaion
Prediction results saved in 'prediction_coco_format.json'
Loading and preparing results...
DONE (t=0.01s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=2.87s).
Accumulating evaluation results...
DONE (t=0.60s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.008
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.004
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.006
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.006
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.006
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
Epoch 45/50
1031/1562 [==================>...........] - ETA: 52s - loss: 4.2126
IOPub message rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_msg_rate_limit`.

Current values:
NotebookApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

In [53]:

```
loss = history.history['loss']

plt.plot(loss, label='Training Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,20.0])
plt.title('Training Loss')
```

```python
plt.xlabel('epoch')
plt.show()
```



과제: Model 개선

실습파일에 제공된 모델은 Detection layer 5개, 3가지 종류의 Aspect ratio를 가진 Anchor box로 구성된다. Model의 Detection 성능을 향상시키기 위해 모델에 아래 사항을 반영하라.

1. 1개 Detection Layer 추가: Layer width = 28인 Detection Layer를 2번째 Bottleneck Block 뒤에 추가하여라.
2. 다양한 Anchor box: Anchor box의 aspect ratio가 1, 2, 1/2만 사용되고 있다. SSD 논문에서 사용한 Aspect ratio들을 활용하기 위해 아래의 변화를 반영해 보자.
   - 모든 Detection layer에 대하여 현재 Layer의 s값과 다음 Layer의 s값을 곱한뒤 제곱근을 취한 값을 곱해주도록 변경하여라.
   - 1번 항목을 완료하여 6개의 Detection layer가 있는 상태에서, 2,3,4번째 Layer에 대하여 aspect ratio=3, 1/3을 추가한다.
     SSD 논문에서 해당 내용은 아래와 같다.

for prediction. The scale of the default boxes for each feature map is computed as:

$$s_k = s_{\text{min}} + \frac{s_{\text{max}} - s_{\text{min}}}{m-1}(k-1), \quad k \in [1, m] \tag{4}$$

where s_{min} is 0.2 and s_{max} is 0.9, meaning the lowest layer has a scale of 0.2 and the highest layer has a scale of 0.9, and all layers in between are regularly spaced. We impose different aspect ratios for the default boxes, and denote them as $a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$. We can compute the width ($w_k^a = s_k\sqrt{a_r}$) and height ($h_k^a = s_k/\sqrt{a_r}$) for each default box. For the aspect ratio of 1, we also add a default box whose scale is $s'_k = \sqrt{s_k s_{k+1}}$, resulting in 6 default boxes per feature map location. We set the center

1. 현재 Localization loss는 L2 loss가 사용되고 있다. 이를 Smooth L1 Loss(Huber Loss)로 바꾸어 적용하여라. Smooth L1 Loss는 아래와 같다. delta=1을 적용하여라.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

1. Data augmentation: 이미지에 Random horizontal flip을 적용하여라. Canvas 전체를 flip하지 말고, CIFAR10 이미지를 flip한 후, canvas에 랜덤하게 배치한다.

위의 사항을 반영하여 수정하고 보고서를 작성하여 수정된 Jupyter Notebook과 함께 제출하여라. 보고서에는 아래 내용이 담겨야 한다. 조교가 트레이닝을 수행해본 결과 트레이닝 중에 mAP의 등락이 있는 것이 발견되었다. Data의 특성에 의한 것으로 판단되니, 이러한 현상이 발견되더라도 당황하지 말고 트레이닝을 진행해도 된다.

- 원본 실습 코드에서 어느 부분을 어떻게 수정하였는지에 대한 설명
- 수정된 모델을 Training하고 최고 성능을(mAP)를 기록

In [ ]:

```

```