In [21]:

```
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm
from random import shuffle
```

In [22]:

```
MAIN_DIR = os.getcwd()
working_dir = os.path.join(MAIN_DIR,"train")


os.chdir(working_dir)
```

In [23]:

```
def mirroring_y(data):
    list_of_lists = [['xmin',19],['xmax',21]]
    min_max_list = []
    for v in list_of_lists:
        sep_1 = v[0]
        a = data[v[1]].split(sep_1)[0]
        b = data[v[1]].split(sep_1)[1]
        c = data[v[1]].split(sep_1)[2]
        sep_2 = '<'
        b_1 = b.split(sep_2)[0]
        b_2 = b.split(sep_2)[1]
        b_1_1 = '>'
        b_1_2 = str(227 - int(b_1[1:]))
        changed_string = a + sep_1 + b_1_1 + b_1_2 + sep_2 + b_2
+ sep_1 + c
        min_max_list.append(changed_string)
    return list((min_max_list))
```

In [24]:

```
def make_flipping_y():
    for i in tqdm([x for x in os.listdir() if x.split('.')[1] ==
'jpg' ]):
        object_name = i.split('.')[0]
        path = os.path.join(working_dir,i) # Setting the directo
ry of image which is going to be read.
        img = cv2.imread(path,1) # Reading image
        img = cv2.flip( img, 1 ) # Mirroring image wrt y axis
        cv2.imwrite(object_name + '_y' + '.jpg',img) # Saving fl
ipped image
        with open(str(object_name + '.xml')) as f: # Reading the
.xml file of image
            content = f.readlines()
        [content[19],content[21]] = mirroring_y(content)
        with open(object_name + '_y' + '.xml', "w") as f:
            for s in content:
                f.write(str(s))

make_flipping_y()
```

```
100%|██████████| 7824/7824 [00:29<00:00, 262.12it/s]
```

In [25]:

```
def mirroring_x(data):
    list_of_lists = [['ymin',20],['ymax',22]]
    min_max_list = []
    for v in list_of_lists:
        sep_1 = v[0]
        a = data[v[1]].split(sep_1)[0]
        b = data[v[1]].split(sep_1)[1]
        c = data[v[1]].split(sep_1)[2]
        sep_2 = '<'
        b_1 = b.split(sep_2)[0]
        b_2 = b.split(sep_2)[1]
        b_1_1 = '>'
        b_1_2 = str(227 - int(b_1[1:]))
        changed_string = a + sep_1 + b_1_1 + b_1_2 + sep_2 + b_2
+ sep_1 + c
        min_max_list.append(changed_string)
    return list(min_max_list)
```

In [26]:

```
def make_flipping_x():
    for i in tqdm([x for x in os.listdir() if x.split('.')[1] ==
'jpg' ]):
        object_name = i.split('.')[0]
        path = os.path.join(working_dir,i)# Setting the director
y of image which is going to be read.
        img = cv2.imread(path,1) # Reading image
        img = cv2.flip( img, 0 ) # Mirroring image wrt y axis
        cv2.imwrite(object_name + '_x' + '.jpg',img) # Saving fl
ipped image
        with open(str(object_name + '.xml')) as f: # Reading the
.xml file of image
            content = f.readlines()
        [content[20],content[22]] = mirroring_x(content)
        with open(object_name + '_x' + '.xml', "w") as f:
            for s in content:
                f.write(str(s))

make_flipping_x()
```

```
100%|██████████| 14400/14400 [00:49<00:00, 292.66it/
s]
```

In [17]:

```
def add_noise():
    for i in tqdm([x for x in os.listdir() if x.split('.')[1] ==
'jpg' ]):
        object_name = i.split('.')[0]
        path = os.path.join(working_dir,i)
        img = cv2.imread(path,1)
        noise  = 2 * img.std() * np.random.random(img.shape)
        img = img + noise
        cv2.imwrite(object_name + '_noised' + '.jpg',img)
        with open(str(object_name + '.xml')) as f:
            content = f.readlines()
        with open(object_name + '_noised' + '.xml', "w") as f:
            for s in content:
                f.write(str(s))

add_noise()
```

```
100%|██████████| 4248/4248 [00:26<00:00, 162.71it/s]
```

In [30]:

```
def xml_reader(data):
    list_of_lists = [['xmin',19],['ymin',20],['xmax',21],['ymax'
,22]]
    min_max_list = []
    for v in list_of_lists:
        sep_1 = v[0]
        a = data[v[1]].split(sep_1)[0]
        b = data[v[1]].split(sep_1)[1]
        c = data[v[1]].split(sep_1)[2]
        sep_2 = '<'
        b_1 = b.split(sep_2)[0]
        b_2 = b.split(sep_2)[1]
        b_1_1 = '>'
        b_1_2 =  int(b_1[1:])
        val = b_1_2
        min_max_list.append(val)
    if min_max_list[0] > min_max_list[2]:
        min_max_list[0], min_max_list[2] = min_max_list[2], min_
max_list[0]
    if min_max_list[1] > min_max_list[3]:
        min_max_list[1], min_max_list[3] = min_max_list[3], min_
max_list[1]
    return min_max_list
```

In [31]:

```
def create_data():
    training_data = []
    for i in tqdm([x for x in os.listdir() if x.split('.')[1] ==
'jpg' ]):
        object_name = i.split('.')[0]
        object_name_updated = object_name.split('_')[0]
        if object_name_updated =='ball':
            output_vector = [1,0,0]
        elif object_name_updated =='book':
            output_vector = [0,1,0]
        #elif object_name_updated =='car':
        #    output_vector = [0,0,1,0]

        else:
            output_vector = [0,0,1]
        path = os.path.join(working_dir,i)
        img = cv2.imread(path,1)
        with open(str(object_name + '.xml')) as f:
            content = f.readlines()
        locations = xml_reader(content)
        training_data.append([np.array(img),np.array(output_vect
or),np.array(locations)])
    shuffle(training_data)
    return training_data
```

In [29]:

```
data = create_data()
os.chdir(MAIN_DIR)
np.save('object_localization.npy', data,allow_pickle=True)
```

```
100%|██████████| 26496/26496 [00:50<00:00, 528.41it/
s]
```

In [ ]:

In [ ]: