Object Identification in Starlight Images

1. Introduction

Working on object identification in starlight images using yolov7 engine. The dataset used is xView dataset available on Kaggle. It has 846 train images and 281 images for validation. It also contains a .geojson file containing all the label information of every image.

2. Data Cleaning and Pre-processing

First, we loaded all the images in a separate working folder. We convert them to 'L.' 'L' is a single channel image - normally interpreted as greyscale. It means that is just stores the Luminance. It is very compact, but only stores a greyscale, not colour.

[6]:
```
images = glob('/kaggle/input/xview-dataset/train_images/train_images/*')
for img in tqdm(images):
    image = Image.open(img).convert('L')
    name = img.split('/')[-1].split('.')[0]
    image.save('images/'+ name +'.tiff')
```

```
100%|██████████| 846/846 [05:53<00:00,  2.39it/s]
```

[7]:
```
images = glob('/kaggle/input/xview-dataset/val_images/val_images/*')
for img in tqdm(images):
    image = Image.open(img).convert('L')
    name = img.split('/')[-1].split('.')[0]
    image.save('val-img/'+ name +'.tiff')
```

```
100%|██████████| 281/281 [02:01<00:00,  2.32it/s]
```

We load the geojson file and try to convert it to labels readable for yolov7 engine.

[8]:
```
f = open('/kaggle/input/xview-dataset/train_labels/xView_train.geojson')

# returns JSON object as
# a dictionary
data = json.load(f)
```

[9]:
```
classes_dict = {
    'small_vehicle':[17,18,36],
    'medium_vehicle':[20,34,37],
    'large_vehicle':[21,23,25,26,27,28,32,60,62,63,64,65,66,25],
    'bus':[19],
    'double_trailer_truck':[],
    'container':[91,35,57],
    'heavy_equipment':[56,53,59,61,29],
    'pylon':[93],
    'small_aircraft':[12,15,11],
    'large_aircraft':[11,13],
    'small_vessel':[41,42,47],
    'medium_vessel':[44,50,40],
    'large_vessel':[45,49,51,52]
}
```

In this we are creating a dictionary where all the labels of an image are stored together. The points are normalized and negative labels are dealt with.

[16]:
```
from tqdm import tqdm
import cv2
import os
jsons_1 = {}
imag = list()

for i in tqdm(data['features']):
    catagory_num = i['properties']['type_id']
    class_mafat = get_class(catagory_num)
    if class_mafat != 'err':
        points = i['properties']['bounds_imcoords']
        img_id = i['properties']['image_id']
        if img_id not in imag:
            if(os.path.exists('/kaggle/input/xview-dataset/train_images/train_images/' + img_id) == 0):
                continue
            else:
                imag.append(img_id)
                img = cv2.imread("/kaggle/input/xview-dataset/train_images/train_images/"+ img_id)
                num_rows, num_cols = img.shape[:2]
        new_json = {
            'pixels':create_points(points, num_rows, num_cols),
            'type': classes_dict1.get(class_mafat)
        }
        if img_id not in jsons_1.keys():
            jsons_1[img_id] = []
        jsons_1[img_id].append(new_json)
```

```
100%|██████████| 601937/601937 [02:57<00:00, 3398.80it/s]
```

Now we save those labels in different text files with their corresponding image name as the file name.

```
for i in tqdm(jsons_1.keys()):
    path = 'labels/' + i.split('.')[0] + '.txt'
    f = open(path, "w")
    for k in jsons_1[i]:
        f.write(str(k['type']) + ' ' + k['pixels'] +'\n')
    f.close()
```

3. Preparing yaml file

Now, yolov7 needs input in a different manner. For yolov7, a .yaml is created with all the basic information like the path to train and validation images and the number of classes and the name of those classes.

[23]:
```
data_yaml = dict(
    train = '/kaggle/working/images',
    val = '../val-img',
    nc = 13,
    names= ['small_vehicle','medium_vehicle','large_vehicle','bus','double_trailer_truck','container','heav
)

# Note that I am creating the file in the yolov5/data/ directory.
with open('data3.yaml', 'w') as outfile:
    yaml.dump(data_yaml, outfile, default_flow_style=True)
```

4. Training the Model

Yolov7 requires a good level GPU for training. Due to limited resources, I have taken image size to be 256 and the number of epochs to 5. Weights are initialized to ‘yolov7-tiny.pt.’

```
!python train.py --img 256 --batch 32 --epochs 5 --data data3.yaml  --weights 'yolov7-tiny.pt'
```

```
W&B disabled.
/opt/conda/lib/python3.7/site-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming release, it will be
required to pass the indexing argument. (Triggered internally at /usr/local/src/pytorch/aten/src/ATen/native/TensorShape.cpp:319
0.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
train: Scanning '/kaggle/working/labels.cache' images and labels... 792 found, 5
val: Scanning '../val-img.cache' images and labels... 0 found, 281 missing, 0 em

autoanchor: Analyzing anchors... anchors/target = 0.02, Best Possible Recall (BPR) = 0.0098. Attempting to improve anchors, plea
se wait...
autoanchor: WARNING: Extremely small objects found. 169988 of 172526 labels are < 3 pixels in size.
autoanchor: Running kmeans for 9 anchors on 18865 points...
autoanchor: thr=0.25: 0.9935 best possible recall, 3.81 anchors past thr
autoanchor: n=9, img_size=256, metric_all=0.245/0.496-mean/best, past_thr=0.396-mean: 2,1,  1,3,  3,1,  3,2,  2,4,  5,2,  5,4,
10,10,  22,21
autoanchor: Evolving anchors with Genetic Algorithm:: 100%|| 1000/1000 [00:15<0
autoanchor: thr=0.25: 0.9935 best possible recall, 3.81 anchors past thr
autoanchor: n=9, img_size=256, metric_all=0.245/0.496-mean/best, past_thr=0.396-mean: 2,1,  1,3,  3,1,  3,2,  2,4,  5,2,  5,4,
10,10,  22,21
autoanchor: New anchors saved to model. Update model *.yaml to use these anchors in the future.

      0/4    0.495G    0.1518  0.004714   0.04636    0.2028        43       256
               Class      Images      Labels           P           R      mAP@.5
                 all         281           0           0           0           0           0
      1/4     1.48G    0.1379  0.003617   0.04496    0.1865        47       256
               Class      Images      Labels           P           R      mAP@.5
                 all         281           0           0           0           0           0
      2/4     1.48G    0.1309  0.003177   0.04247    0.1766        97       256
               Class      Images      Labels           P           R      mAP@.5
                 all         281           0           0           0           0           0
      3/4     1.48G    0.1284  0.002634   0.04012    0.1712        23       256
               Class      Images      Labels           P           R      mAP@.5
                 all         281           0           0           0           0           0
      4/4     1.48G    0.1297  0.002596   0.04029    0.1726        66       256
               Class      Images      Labels           P           R      mAP@.5
                 all         281           0           0           0           0           0
```

5. Issues with training

This was one of the most messiest dataset I have worked on. Labels file is not proper and requires a lot of pre-processing to come into effect. Also labels for all the images are not provided. Out of 846, labels for only 729 images are provided and labels for validation images are not provided. These issues factor in with the accuracy of the model which comes out to be very low.

6. Detection and Prediction

[29]:
```
!python detect.py --weights yolov7-tiny.pt --img-size 256 --source /kaggle/input/xview-dataset/train_images
```

```
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.25, device='', exist_ok=False, img_size=256, iou_thres=
0.45, name='exp', no_trace=False, nosave=False, project='runs/detect', save_conf=False, save_txt=False, source='/kaggle/input/xv
iew-dataset/train_images/train_images/100.tif', update=False, view_img=False, weights=['yolov7-tiny.pt'])
Fusing layers...
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!

/opt/conda/lib/python3.7/site-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming release, it will be
required to pass the indexing argument. (Triggered internally at /usr/local/src/pytorch/aten/src/ATen/native/TensorShape.cpp:319
0.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Done. (5.1ms) Inference, (0.4ms) NMS
 The image with the result is saved in: runs/detect/exp2/100.tif
Done. (0.235s)
```

One of the image is taken as a test image and labels are predicted and objects are detected. But due to low accuracy, results aren't good and here is the predicted image.

```
import matplotlib.pyplot as plt

# read an image
img = cv2.imread("/kaggle/working/yolov7/runs/detect/exp2/100.tif")

# write it in a new format
plt.imshow(img)
```

[33]:
```
<matplotlib.image.AxesImage at 0x7f2ace655790>
```