Title : Child Seat Localization

Goal Build AI solution for child seat localization in the passenger vehicle

Problem statement Localization of the child seat in the passenger vehicle enable the buisness to high-level services and usecases.

Data set The data set shall be https://sviro.kl.dfki.de/data/ or any appropriate data set. You will have to register with an account here. It takes a day to get the approval so start with registering yourself first. It's a synthetic dataset with different images including RGB, Infrared and Bounding Box ground truths.

Solution scope We propose to use Jupyter Notebook with proper markdown text to explain your thought process and logic.
Proposed software solution shall have following components:

```
1. Instrument the data set – Exploratory Analysis

2. State or use prior work from academic / industry

3. Use appropriate state of the art AI algorithm (preferably deep learning based)

4. Enhance the data with synthetic images based on need and your algorithm (optional)

5. Reasonable accuracy on the given scope of time to develop

6. Rationales for the current Approach to enhance

7. Use any appropriate open source libraries and framework

8. Foucs more on the algorithm than the overall application

9. Apply coding standards and best practices (We emphasize on clean code)

10. Make sure to give proper credits at the end (For example, citation for the dataset used in
this challenge)
```

Delivery

```
• The solution shall be presented github / gitlab code with optional supported document in
email.

• Execute all the cells of your notebook and export it to PDF. Include it in your source code
repository.

• Prepare a 30 minute presentation on your solution describing challenges faced, choice of
algorithms, results of your analysis and illustrations.
```

In [1]:

```
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt
from glob import glob
from tqdm.notebook import tqdm
import os
import shutil
import random
import json

from utils.general import get_coords, extract_from_txt, convert_to_yolov5, generate_json_coco, save_one_json, u
from utils.plots import plot_one_box, plot_image_cropped_new

%matplotlib inline
```

```
Bad key "text.kerning_factor" on line 4 in
/home/jm/anaconda3/envs/yolo/lib/python3.7/site-packages/matplotlib/mpl-data/stylelib/_classic_test_patch.mplst
yle.
You probably need to get an updated matplotlibrc file from
https://github.com/matplotlib/matplotlib/blob/v3.1.2/matplotlibrc.template
or from the matplotlib source distribution
```

Dataset exploratory analysis

The dataset is sintetic, meaning made of image that were rendered based on real models and is divided in 10 example of cars and 4 classes, being infant seat, child seat, person and every day object. To compose the dataset I have downloaded only the RGB images, and chose the first 6 cars, summing up a total of 12000 images. From those, 10%, 1200 images were randomly selected and moved to validation. The test set is already separeted within each zip file, and it was added to the test folder.

The dataset is supposed to be separated in train, val and test, in a folder named images, inside the dataset folder, and the labels, which is in the xyxy format is supposed to be in a folder named labels_xyxy, inside the dataset folder, also in the train, val, test fashion, matching the names of the files '.png' and '.txt'.

In [2]:

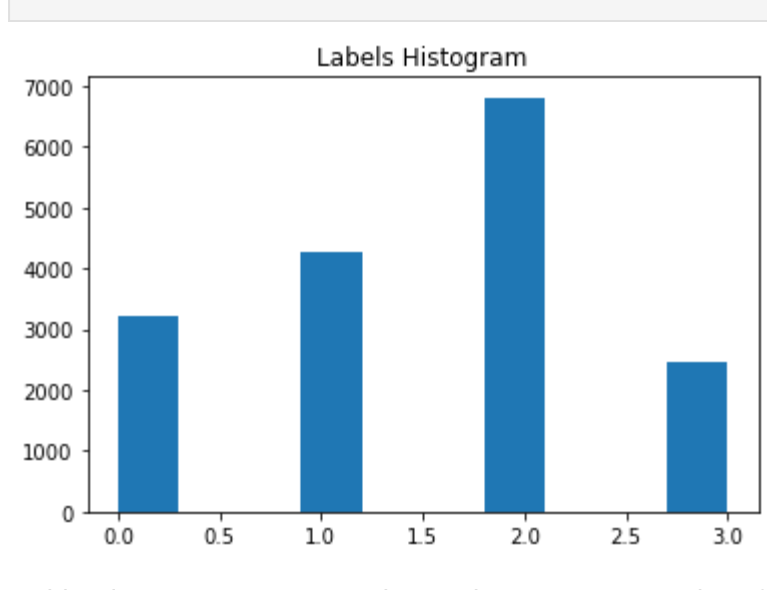
```
# Support paths and label dictionary
image_path = './dataset/images/'
label_path = './dataset/labels_xyxy/'

labels_dict = {0: 'infant seat',
               1: 'child seat',
               2: 'person',
               3: 'everyday object'}

# All dataset files, for visualization
all_imgs = glob('./dataset/images/train/*.*')
all_labels = glob('./dataset/labels_xyxy/train/*.*')
```

In [3]:

```
# Get all the labels to check for quantity
label_hist = list()
for label in all_labels:
    info = get_coords(label)
    try:
        for l in info[:,0].tolist():
            label_hist.append(l-1)
    except:
        continue
```

In [4]:

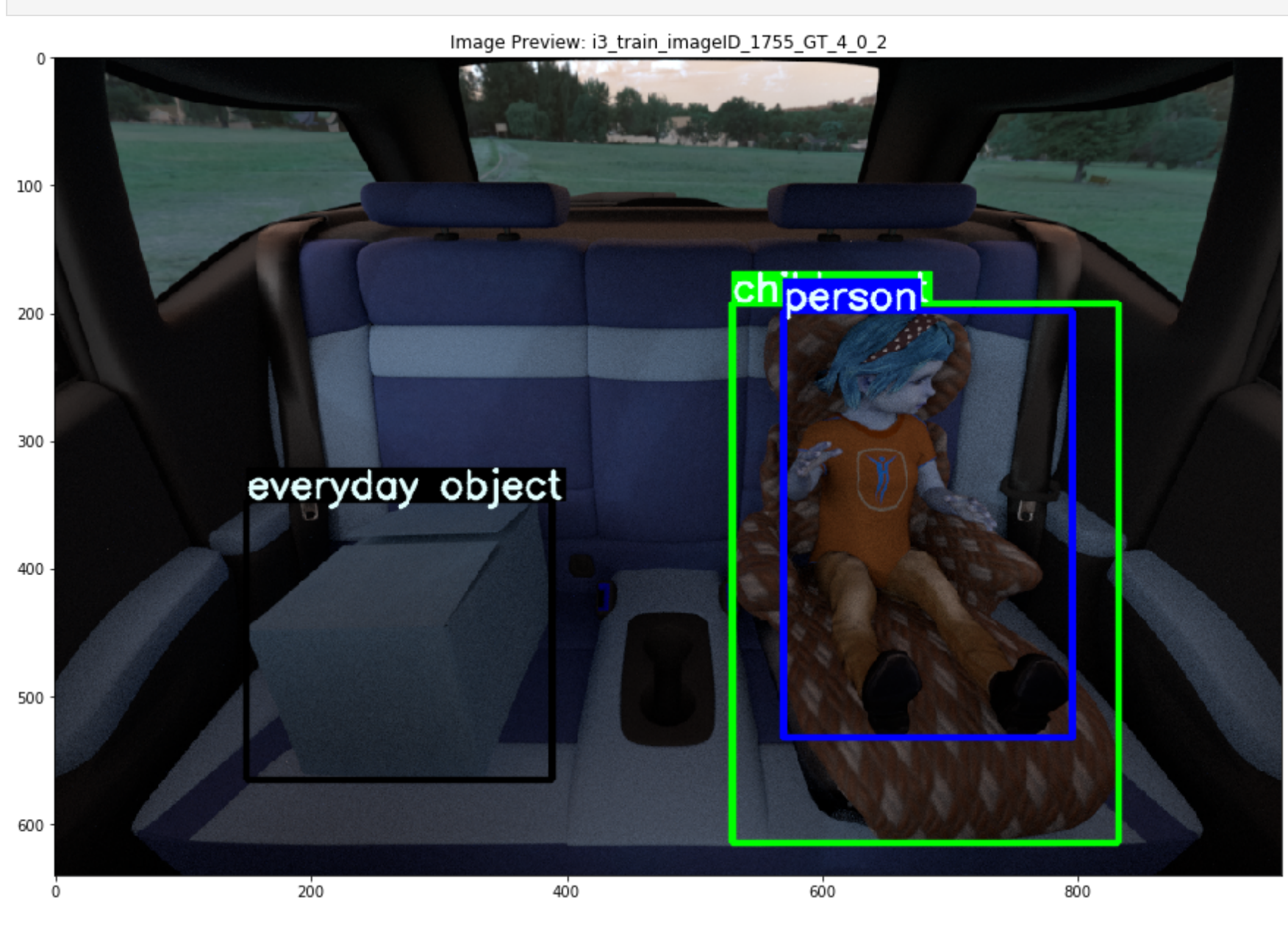
```
# Plot labels histogram
plt.hist(label_hist)
plt.title('Labels Histogram')
plt.show()
```



In this Histogram we can see that we have more examples of person, and a lot less of every day objects, which shows an inbalanced dataset.

In [5]:

```
# Get a random image from train set and plot to check

# Get random image and name
img_num = np.random.choice(len(all_imgs))
img_name = all_imgs[img_num].split('/')[-1].split('.')[0]

# Open the image with OpenCV
pic = cv2.imread(image_path + 'train/' + img_name + '.png')

# Get image box coordenates
box_coords_true = np.array(get_coords(label_path + 'train/' + img_name + '.txt'))
box_coords_true[:,0] = box_coords_true[:,0]-1

# Plot image
returned_true = plot_image_cropped_new(pic, box_coords_true)
plt.figure(figsize=(15,15))
plt.title('Image Preview: {}'.format(img_name))
plt.imshow(returned_true)
plt.show()
```



Related work on SVIRO

The Academic related on SVIRO described in "SVIRO: Synthetic Vehicle Interior Rear Seat Occupancy Dataset and Benchmark", was more on the CLassification task where it classsified the cropped images, based on 3 parts of the back of the vehicle, and also semantic segmentation, where it used a fine tuned Mask RCNN with a ResNet-50 backbone to achieve the task. For my project, to reach the goal of object detection for the child seat, I chose to use the Yolo approach, where now it is on version v5.

State of the Art in Object Detection

With the advances in the usage of Transformers, which had a great impact with Natural Language Processing, the same idea is now used in Computer Vision, and Facebook AI have developed a new Vision Transformers, called Detectron2, which now holds the SOTA in Object Detection, prior works are the You Only Look Once, or YOLO for short, that had the best scores on COCO dataset, with also a great inference time and FasterRCNN, that differently from those single shot detection methods, it uses a Region Proposal Network for the bounding box regions.

For this project, as I stated before, I decided to use the YOLO algorithm as I am familiar with and has a great performance and can be trained in smaller GPUs or even CPUs with the smallest model. I also have experience with the FasterRCNN model with a ResNet50 or mobilenetV2 backbone, but as it takes longer to converge, Yolo seems the best option, as also the available code is well maintained and complete, giving you several metrics while training and also image augmentations.

Yolo algorithm - Training the model with the Yolov5 repository

The algorithm expects the .yaml file inside the data to have the following informations:

```
train: ./dataset/images/train/
val:  ./dataset/images/val/
test: ./dataset/images/test/

# number of classes
nc: 4

# class names
names: ["infant seat", "child seat", "person", "everyday object"]
```

Also the labels, had to be in the xywh format, instead of the xyxy that is the current annotation format.

In [11]:

```
# Preparing the dataset for the yolo requirements - xyxy to xywh

# Convert and save the annotations
task = 'train/'
all_txt = glob('./dataset/labels_xyxy/' + task + '*.txt')
for txt in tqdm(all_txt, desc='Converting Train'):
    info_dict = extract_from_txt(txt, task=task)
    convert_to_yolov5(info_dict, task)

# Convert and save the annotations
task = 'val/'
all_txt = glob('./dataset/labels_xyxy/' + task + '*.txt')
for txt in tqdm(all_txt, desc='Converting Val'):
    info_dict = extract_from_txt(txt, task=task)
    convert_to_yolov5(info_dict, task)

# Convert and save the annotations
task = 'test/'
all_txt = glob('./dataset/labels_xyxy/' + task + '*.txt')
for txt in tqdm(all_txt, desc='Converting Test'):
    info_dict = extract_from_txt(txt, task=task)
    convert_to_yolov5(info_dict, task)
```

In [12]:

```
# Generate Json for COCO Eval - to be able to use the same metrics as it was used by the SVIRO team

# Annotations JSON generation
all_dataset = glob('./dataset/images/*/*.png')

json_dict = generate_json_coco(all_dataset)

anno_json = 'annotations.json'  # predictions json
print(f'\nSaving {anno_json}...')
with open(anno_json, 'w') as f:
    json.dump(json_dict, f)
```

```
Generating Json: 100%|██████████| 9882/9882 [00:09<00:00, 1090.74it/s]
Saving annotations.json...
```

Training the model with a resized image of 640x640, with the pre trained weights for the small yolo, for 1 epoch. (I have trained a model in the Google Colab, to be able to use the GPU and train for more Epochs.)

The Jupyter notebook is also in the repository named Yolov5_bosh_colab_train.ipynb

In [13]:

```
!python train.py --img 640 --cfg models/yolov5s.yaml --hyp data/hyps/hyp.scratch.yaml --batch 16 --epochs 6 --d
```

```
Bad key "text.kerning_factor" on line 4 in
/home/jm/anaconda3/envs/yolo/lib/python3.7/site-packages/matplotlib/mpl-data/stylelib/_classic_test_patch.mplst
yle.
You probably need to get an updated matplotlibrc file from
https://github.com/matplotlib/matplotlib/blob/v3.1.2/matplotlibrc.template
or from the matplotlib source distribution
train: weights=yolov5s.pt, cfg=models/yolov5s.yaml, data=/data/cust_detect.yaml, hyp=data/hyps/hyp.scratch.yam
l, epochs=6, batch_size=16, imgsz=640, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False,
 evolve=None, bucket=, cache_images=False, image_weights=False, device=, multi_scale=False, single_cls=False, a
dam=False, sync_bn=False, workers=8, project=runs/train, entity=None, name=cust_bosh, exist_ok=False, quad=Fals
e, linear_lr=False, label_smoothing=0.0, upload_dataset=False, bbox_interval=-1, save_period=-1, artifact_alias
=latest, local_rank=-1
github: skipping check (not a git repository), for updates see https://github.com/ultralytics/yolov5
YOLOv5 🚀 2021-7-28 torch 1.8.1+cu102 CPU

hyperparameters: lr0=0.01, lrf=0.2, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.
8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=2.
0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.2, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud
=0.0, fliplr=0.5, mosaic=0.8, mixup=0.3, copy_paste=0.0
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 🚀 runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=4

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1    156928  models.common.C3                        [128, 128, 3]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  1    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     24273  models.yolo.Detect                      [4, [[10, 13, 16, 30, 33, 23], [30,
 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
Model Summary: 283 layers, 7071633 parameters, 7071633 gradients, 16.4 GFLOPs

Transferred 354/362 items from yolov5s.pt
Scaled weight_decay = 0.0005
optimizer: SGD with parameter groups 59 weight, 62 weight (no decay), 62 bias
albumentations: version 1.0.3 required by YOLOv5, but version 0.5.2 is currently installed
train: Scanning 'dataset/labels/train' images and labels...7182 found, 0 missing
train: New cache created: dataset/labels/train.cache
val: Scanning 'dataset/labels/val' images and labels...1200 found, 0 missing, 3
val: New cache created: dataset/labels/val.cache
Plotting labels...

autoanchor: Analyzing anchors... anchors/target = 4.03, Best Possible Recall (BPR) = 1.0000
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to runs/train/cust_bosh
Starting training for 6 epochs...

     Epoch   gpu_mem       box       obj       cls    labels  img_size
       0/5        0G   0.06216  0.004942  0.005004        69       640: 100%|█|
               Class     Images     Labels          P          R     mAP@.5 mAP@
                 all       1200       2771      0.536      0.744      0.565      0.377

     Epoch   gpu_mem       box       obj       cls    labels  img_size
       1/5        0G   0.03883  0.003724  0.003086        84       640: 100%|█|
               Class     Images     Labels          P          R     mAP@.5 mAP@
                 all       1200       2771      0.574      0.841      0.767      0.512

     Epoch   gpu_mem       box       obj       cls    labels  img_size
       2/5        0G   0.03444  0.003194  0.002595        67       640: 100%|█|
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@WARNING: NMS time limit 10.0s e
xceeded
               Class     Images     Labels          P          R     mAP@.5 mAP@^C
               Class     Images     Labels          P          R     mAP@.5 mAP@
Traceback (most recent call last):
  File "train.py", line 583, in <module>
    main(opt)
  File "train.py", line 482, in main
    train(opt.hyp, opt, device)
  File "train.py", line 354, in train
    compute_loss=compute_loss)
  File "/home/jm/anaconda3/envs/yolo/lib/python3.7/site-packages/torch/autograd/grad_mode.py", line 27, in deco
rate_context
    return func(*args, **kwargs)
  File "/home/jm/Projetos/Bosh/Yolov5_cust_obj/val.py", line 178, in run
    out = non_max_suppression(out, conf_thres, iou_thres, labels=lb, multi_label=True, agnostic=single_cls)
  File "/home/jm/Projetos/Bosh/Yolov5_cust_obj/utils/general.py", line 570, in non_max_suppression
    i = torchvision.ops.nms(boxes, scores, iou_thres)  # NMS
  File "/home/jm/anaconda3/envs/yolo/lib/python3.7/site-packages/torchvision/ops/boxes.py", line 36, in nms
    return torch.ops.torchvision.nms(boxes, scores, iou_threshold)
KeyboardInterrupt
```

Validation of the model, using the images in the val folder, with the previous trained model, saving the predicted values in a txt toi be used in the COCOtools eval.

In [15]:

```
!python val.py --img 640 --batch 16 --data /data/cust_detect.yaml --weights runs/train/cust_bosh/weights/best.p
```

```
Bad key "text.kerning_factor" on line 4 in
/home/jm/anaconda3/envs/yolo/lib/python3.7/site-packages/matplotlib/mpl-data/stylelib/_classic_test_patch.mplst
yle.
You probably need to get an updated matplotlibrc file from
https://github.com/matplotlib/matplotlib/blob/v3.1.2/matplotlibrc.template
or from the matplotlib source distribution
val: data=./data/cust_detect.yaml, weights=['runs/train/cust_bosh/weights/best.pt'], batch_size=16, imgsz=640,
 conf_thres=0.001, iou_thres=0.6, task=val, device=, single_cls=False, augment=False, verbose=False, save_txt=T
rue, save_hybrid=False, save_conf=True, save_json=False, project=runs/val, name=val_bosh, exist_ok=False, half=
False
YOLOv5 🚀 2021-7-28 torch 1.8.1+cu102 CPU

Fusing layers...
Model Summary: 224 layers, 7062001 parameters, 0 gradients, 16.4 GFLOPs
val: Scanning 'dataset/labels/val.cache' images and labels... 1200 found, 0 miss
               Class     Images     Labels          P          R     mAP@.5 mAP@
                 all       1200       2771      0.574      0.841      0.767      0.513
         infant seat       1200        553      0.718      0.774      0.784       0.49
          child seat       1200        704      0.401      0.986      0.759      0.591
              person       1200       1110       0.56      0.965      0.914       0.62
     everyday object       1200        404      0.619      0.639       0.61      0.351
Speed: 0.5ms pre-process, 90.3ms inference, 176.0ms NMS per image at shape (16, 3, 640, 640)
Results saved to runs/val/val_bosh2
1200 labels saved to runs/val/val_bosh2/labels
```

Results of the model trained with more images on colab

In [16]:

```
!python val.py --img 640 --batch 16 --data /data/cust_detect.yaml --weights best_colab.pt --task test --name te
```

```
Bad key "text.kerning_factor" on line 4 in
/home/jm/anaconda3/envs/yolo/lib/python3.7/site-packages/matplotlib/mpl-data/stylelib/_classic_test_patch.mplst
yle.
You probably need to get an updated matplotlibrc file from
https://github.com/matplotlib/matplotlib/blob/v3.1.2/matplotlibrc.template
or from the matplotlib source distribution
val: data=./data/cust_detect.yaml, weights=['best_colab.pt'], batch_size=16, imgsz=640, conf_thres=0.001, iou_t
hres=0.6, task=test, device=, single_cls=False, augment=False, verbose=False, save_txt=True, save_hybrid=False,
 save_conf=True, save_json=False, project=runs/val, name=test_bosh_colab, exist_ok=False, half=False
YOLOv5 🚀 2021-7-28 torch 1.8.1+cu102 CPU

Fusing layers...
Model Summary: 224 layers, 7062001 parameters, 0 gradients, 16.4 GFLOPs
test: Scanning 'dataset/labels/test' images and labels...1500 found, 0 missing,
test: New cache created: dataset/labels/test.cache
               Class     Images     Labels          P          R     mAP@.5 mAP@
                 all       1500       3536      0.723      0.785      0.846      0.655
         infant seat       1500        725       0.85      0.578      0.815      0.607
          child seat       1500        841      0.734      0.751      0.779      0.612
              person       1500       1409      0.832      0.933      0.958       0.74
     everyday object       1500        561      0.476      0.879      0.832      0.659
Speed: 0.6ms pre-process, 86.2ms inference, 308.7ms NMS per image at shape (16, 3, 640, 640)
Results saved to runs/val/test_bosh_colab
1500 labels saved to runs/val/test_bosh_colab/labels
```

Visualizing the result via COCO eval tools

In [17]:

```
# read txt and append to jdict
txt_files = glob('./runs/val/test_bosh_colab/labels/*.txt')
```

In [18]:

```
# Opening JSON file
f = open('annotations.json')

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

# Closing file
f.close()
```

In [19]:

```
jdict = []
all_ids = []

for files in txt_files:
    l_list = list()
    box_coords_pred = list()
    r_labels = pd.read_table(files, names=['infos']).values.tolist()
    for b in r_labels:
        label, x, y, w, h, score = list(map(float, b[0].split()))
        x1, y1, x2, y2 = unconvert(640, 640, x, y, w, h)
        l_list.append([label, x1, y1, x2, y2, score])
        if score > 0.5:
            box_coords_pred.append([label, x1, y1, x2, y2])
    save_one_json(np.array(l_list), jdict, files, data_json, all_ids)
```

In [20]:

```
from pycocotools.coco import COCO
from pycocotools.cocoeval import COCOeval

if True and len(jdict):
    pred_json = 'predictions.json'  # predictions json
    print(f'\nEvaluating pycocotools mAP... saving {pred_json}...')
    with open(pred_json, 'w') as f:
        json.dump(jdict, f)

    anno_json = 'annotations.json'  # predictions json
    print(f'\nEvaluating pycocotools mAP... saving {anno_json}...')
    try:  # https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocoEvalDemo.ipynb
        anno = COCO(anno_json)  # init annotations api
        pred = anno.loadRes(pred_json)  # init predictions api
        eval = COCOeval(anno, pred, 'bbox')
        eval.params.imgIds = all_ids  # image IDs to evaluate
        eval.evaluate()
        eval.accumulate()
        eval.summarize()
        map_, map50 = eval.stats[:2]  # update results (mAP@0.5:0.95, mAP@0.5)
    except Exception as e:
        print(f'pycocotools unable to run: {e}')
```

```
Evaluating pycocotools mAP... saving predictions.json...

Evaluating pycocotools mAP... saving annotations.json...
loading annotations into memory...
Done (t=0.06s)
creating index...
index created!
Loading and preparing results...
DONE (t=2.19s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=7.34s).
Accumulating evaluation results...
DONE (t=1.24s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.217
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.286
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.235
 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 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.218
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.396
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.870
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.878
 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 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.878
```

Visualizing some predicted bounding boxes with the respective image

In [37]:

```
# Visualizing the results

# All labels files, for visualization
all_labels = glob('./runs/val/test_bosh_colab/labels/*.*')

# Get a random image from train set and plot to check

# Get random image and name
img_num = np.random.choice(len(all_labels))
img_name = all_labels[img_num].split('/')[-1].split('.')[0]

# Open the image with OpenCV
pic = cv2.imread(image_path + 'test/' + img_name + '.png')
pic = cv2.resize(pic, (640,640))

# Get image box coordenates

label_pred = pd.read_table('./runs/val/test_bosh_colab/labels/' + img_name + '.txt', names=['cls'])
box_coords_pred = list()
for box in label_pred['cls']:
    label, x, y, w, h, score = list(map(float, box.split()))
    x1, y1, x2, y2 = unconvert(640, 640, x, y, w, h)
    if score > 0.3:
        box_coords_pred.append([label, x1, y1, x2, y2])


# Plot image
returned_true = plot_image_cropped_new(pic, box_coords_pred)
plt.figure(figsize=(15,15))
plt.title('Image Preview: {}'.format(img_name))
plt.imshow(returned_true)
plt.show()
```

Confusion Matrix generated by the yolo output

In [ ]: