2024 Coding Challenge: Automated Object Detection and Counting

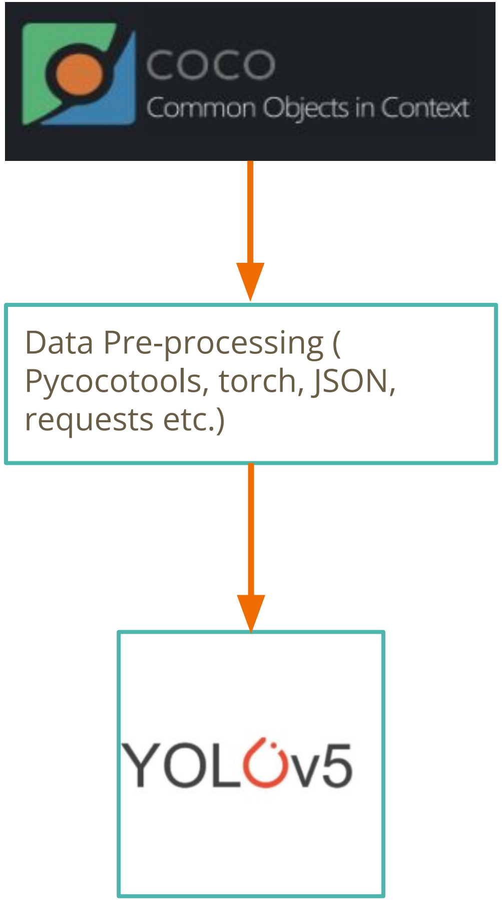
Taïsha Joseph | Github | LinkedIn

June 24, 2024

Objective

- Develop a Python program to count and detect objects from images using computer vision techniques.
- The model should be able to recognize and count the following objects (**person, car, bicycle**) in multiple contexts.

Methodology



Data Preprocessing

- Download training and validation images with annotations from COCO
- Filter images and annotations for the 3 categories of interest using Python library pycocotools
- **Optional:** balance the number of images per classes by:
  - Limiting the number of images per categories based on the category with the minimum number of images
  - Limiting the number per categories based on a predefined number of images (i.e., 500) to reduce training time

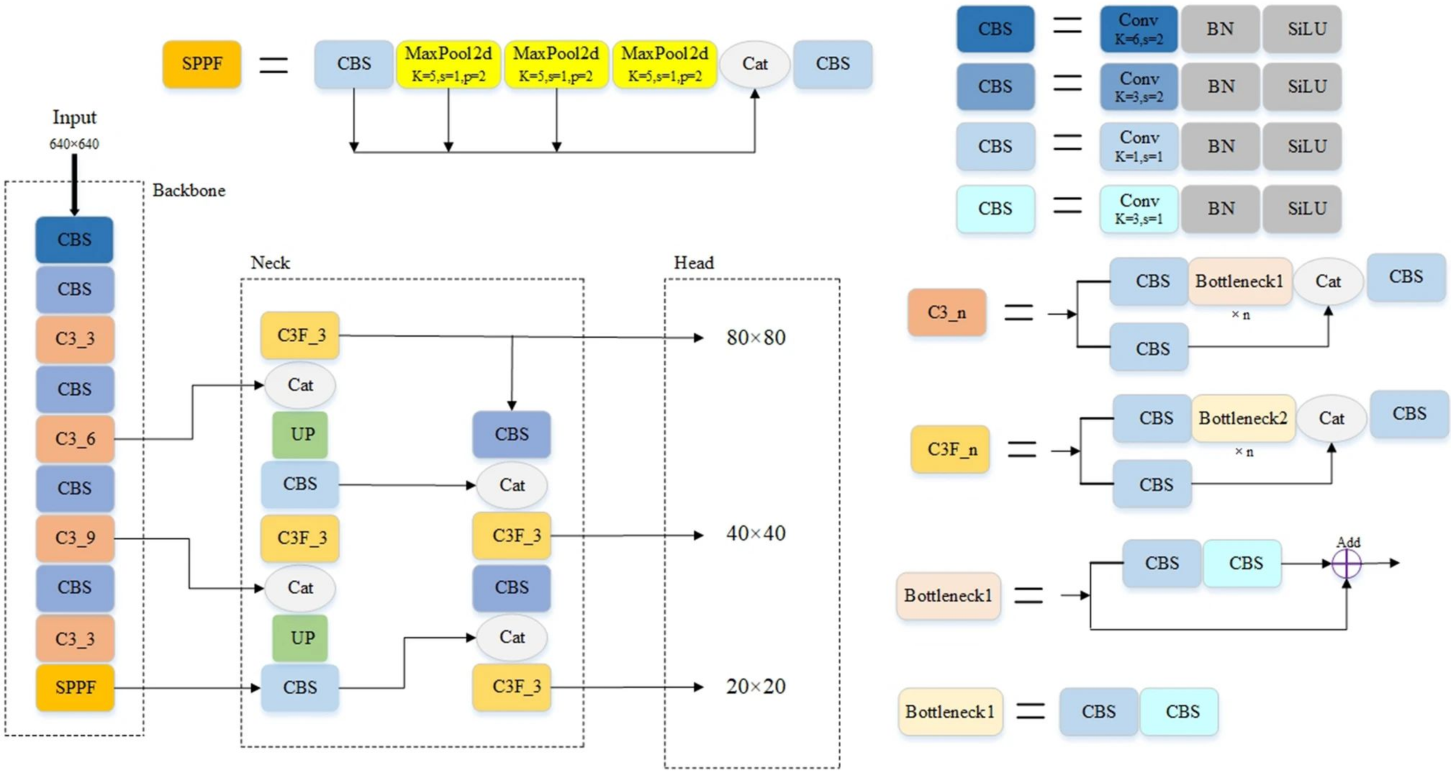
```python
def filter_coco_annotations(input_annotation_file, output_annotation_file, image_dir, output_image_dir, categorie
    coco = COCO(input_annotation_file)
    category_ids = coco.getCatIds(catNms=categories_to_keep)
    image_ids = coco.getImgIds(catIds=category_ids)

    filtered_annotations = []
    filtered_images = []

    ensure_dir_exists(output_image_dir)
    ensure_dir_exists(os.path.dirname(output_annotation_file))

    for img_id in image_ids:
        img_info = coco.loadImgs(img_id)[0]
        ann_ids = coco.getAnnIds(imgIds=img_id, catIds=category_ids)
        anns = coco.loadAnns(ann_ids)

        filtered_images.append(img_info)
        filtered_annotations.extend(anns)

        # Copy image to output directory
        src_img_path = os.path.join(image_dir, img_info['file_name'])
        dst_img_path = os.path.join(output_image_dir, img_info['file_name'])
        shutil.copy(src_img_path, dst_img_path)

    filtered_data = {
        'images': filtered_images,
        'annotations': filtered_annotations,
        'categories': [cat for cat in coco.loadCats(category_ids)]
    }

    with open(output_annotation_file, 'w') as f:
        json.dump(filtered_data, f)
```

Data Preprocessing

- Create indices for labels to be added to data configuration file
- Extract and normalize coordinates for bounding boxes from annotation file for each image according to YOLO format coordinates.
- Export YOLO-specific labels for each image

```
def convert_coco_to_yolo(annotations_file, labels_dir):
    # Load the COCO annotations file
    with open(annotations_file, 'r') as f:
        data = json.load(f)

    # Create a dictionary to map category IDs to category names
    categories = {cat['id']: cat['name'] for cat in data['categories']}

    # Create a dictionary to map category names to YOLO class indices
    category_to_index = {name: index for index, name in enumerate(categories.values())}

    # Ensure the labels directory exists
    os.makedirs(labels_dir, exist_ok=True)

    # Iterate over all annotations in the COCO dataset
    for ann in data['annotations']:
        image_id = ann['image_id']
        category_id = ann['category_id']
        bbox = ann['bbox']
        category_name = categories[category_id]

        # Skip categories that are not in the category_to_index dictionary
        if category_name not in category_to_index:
            continue

        # Get image information to calculate normalized bounding box coordinates
        image_info = next(img for img in data['images'] if img['id'] == image_id)
        image_width = image_info['width']
        image_height = image_info['height']

        # Calculate YOLO format coordinates (normalized)
        x_center = (bbox[0] + bbox[2] / 2) / image_width
        y_center = (bbox[1] + bbox[3] / 2) / image_height
        width = bbox[2] / image_width
        height = bbox[3] / image_height

        # Create the YOLO label string
        yolo_label = f"{category_to_index[category_name]} {x_center} {y_center} {width} {height}\n"

        # Determine the label file path based on the image file name (without extension)
        label_file_path = os.path.join(labels_dir, f"{image_info['file_name'].split('.')[0]}.txt")

        # Append the YOLO label to the label file
        with open(label_file_path, 'a') as label_file:
            label_file.write(yolo_label)
```

YOLOv5 Network Architecture



[Deng et al., *Sci Rep* 2023]

YOLOv5n Architecture

- Learning rate: 0.01
- Loss functions:
    - **Classification loss:**
        - Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss)
        - FocalLoss (variant of cross-entropy that handles class imbalance) if gamma (focusing param) > 0
    - **Bounding Box regression loss:**
        - Complete Intersection over Union (IoU) loss
- Activation function
    - **SiLU** (Swish) Activation Function (default)

```yaml
# Parameters
nc: 3 # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.25 # layer channel multiple
anchors:
  - [10, 13, 16, 30, 33, 23] # P3/8
  - [30, 61, 62, 45, 59, 119] # P4/16
  - [116, 90, 156, 198, 373, 326] # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [
    [-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
    [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
    [-1, 3, C3, [128]],
    [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
    [-1, 6, C3, [256]],
    [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
    [-1, 9, C3, [512]],
    [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
    [-1, 3, C3, [1024]],
    [-1, 1, SPPF, [1024, 5]], # 9
  ]

# YOLOv5 v6.0 head
head: [
    [-1, 1, Conv, [512, 1, 1]],
    [-1, 1, nn.Upsample, [None, 2, "nearest"]],
    [[-1, 6], 1, Concat, [1]], # cat backbone P4
    [-1, 3, C3, [512, False]], # 13

    [-1, 1, Conv, [256, 1, 1]],
    [-1, 1, nn.Upsample, [None, 2, "nearest"]],
    [[-1, 4], 1, Concat, [1]], # cat backbone P3
    [-1, 3, C3, [256, False]], # 17 (P3/8-small)

    [-1, 1, Conv, [256, 3, 2]],
    [[-1, 14], 1, Concat, [1]], # cat head P4
    [-1, 3, C3, [512, False]], # 20 (P4/16-medium)

    [-1, 1, Conv, [512, 3, 2]],
    [[-1, 10], 1, Concat, [1]], # cat head P5
    [-1, 3, C3, [1024, False]], # 23 (P5/32-large)

    [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
  ]
```

Data Augmentations

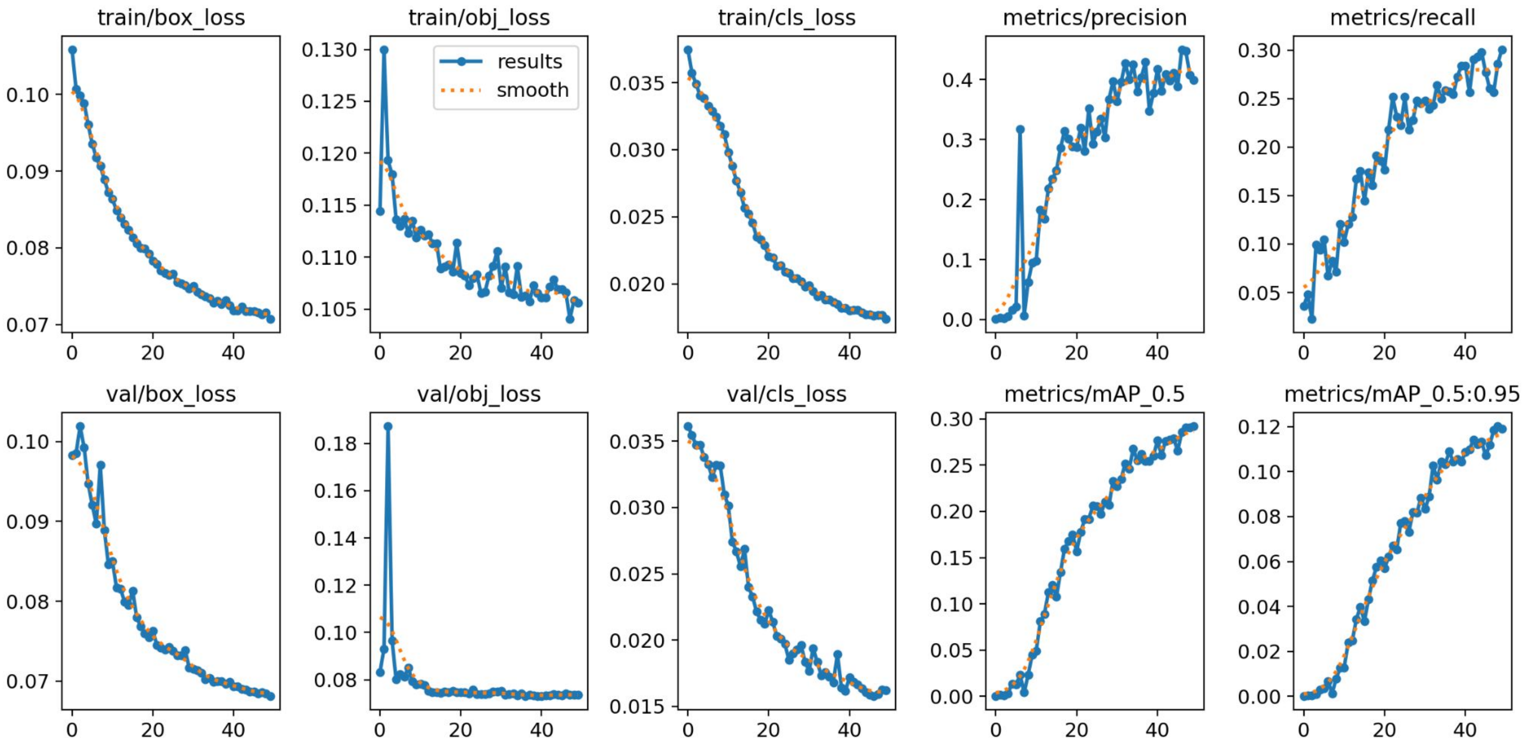
- HSV augmentation (adjust the hue, saturation, and value (brightness) of images)
- Geometric transformations (random rotations, translations, scaling, shearing,etc.).
- Flipping and Mosaic (flipud, fliplr, mosaic)

```yaml
# Ultralytics YOLOv5 🚀, AGPL-3.0 license
# Hyperparameters for low-augmentation COCO training from scratch
# python train.py --batch 64 --cfg yolov5n6.yaml --weights '' --data coco.yaml --img 640 --epochs 300 --linear
# See tutorials for hyperparameter evolution https://github.com/ultralytics/yolov5#tutorials

lr0: 0.01 # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01 # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937 # SGD momentum/Adam beta1
weight_decay: 0.0005 # optimizer weight decay 5e-4
warmup_epochs: 3.0 # warmup epochs (fractions ok)
warmup_momentum: 0.8 # warmup initial momentum
warmup_bias_lr: 0.1 # warmup initial bias lr
box: 0.05 # box loss gain
cls: 0.5 # cls loss gain
cls_pw: 1.0 # cls BCELoss positive_weight
obj: 1.0 # obj loss gain (scale with pixels)
obj_pw: 1.0 # obj BCELoss positive_weight
iou_t: 0.20 # IoU training threshold
anchor_t: 4.0 # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015 # image HSV-Hue augmentation (fraction)
hsv_s: 0.7 # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4 # image HSV-Value augmentation (fraction)
degrees: 0.0 # image rotation (+/- deg)
translate: 0.1 # image translation (+/- fraction)
scale: 0.5 # image scale (+/- gain)
shear: 0.0 # image shear (+/- deg)
perspective: 0.0 # image perspective (+/- fraction), range 0-0.001
flipud: 0.0 # image flip up-down (probability)
fliplr: 0.5 # image flip left-right (probability)
mosaic: 1.0 # image mosaic (probability)
mixup: 0.0 # image mixup (probability)
copy_paste: 0.0 # segment copy-paste (probability)
```

Training Object Detection Model

- 3 attempts with two models:
  1. Run 1:
     a. Trained base YOLOv5n model from scratch for 50 epochs on full COCO 2017 training dataset
     b. Validation using filtered COCO 2017 val dataset
  2. Run 2:
     a. Added dropout regularization layers to improve performance of model on validation set and prevent overfitting.
     b. Increased depth and width of network from model default to 0.5.
     c. Trained modified model for 100 epochs on COCO 2017 training dataset limited by the size of the smallest category to balance the classes.
     d. Validation using filtered COCO 2017 val dataset
  3. Run 3:
     a. Same model architecture as run 2
     b. Maintained balanced classes but limited the number of images in the training set to 500 for each class
     c. Trained modified model for 200 epochs on limited COCO 2017 training dataset
     d. Validation using filtered COCO 2017 val dataset

Evaluation metrics



Inference Results

Inference

In [37]:

```
!python detect.py --weights "runs/train/exp/weights/best.pt" --img 640 --conf 0.25 --source data/images
```

```
detect: weights=['runs/train/exp/weights/best.pt'], source=data/images, data=data/coco128.yaml, imgsz=[640, 640],
conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_csv=False, save_conf=
False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=Fal
se, project=runs/detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=Fals
e, dnn=False, vid_stride=1
YOLOv5 🚀 v7.0-321-g3742ab49 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (NVIDIA L4, 22700MiB)

Fusing layers...
YOLOv5n_mod summary: 157 layers, 1763224 parameters, 0 gradients, 4.1 GFLOPs
WARNING ⚠ NMS time limit 0.550s exceeded
image 1/2 /content/yolov5/data/images/bus.jpg: 640x480 2 persons, 1 car, 1 bicycle, 98.8ms
image 2/2 /content/yolov5/data/images/zidane.jpg: 384x640 (no detections), 103.5ms
Speed: 0.5ms pre-process, 101.1ms inference, 282.3ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp
```

Inference Results

How to Improve Performance?

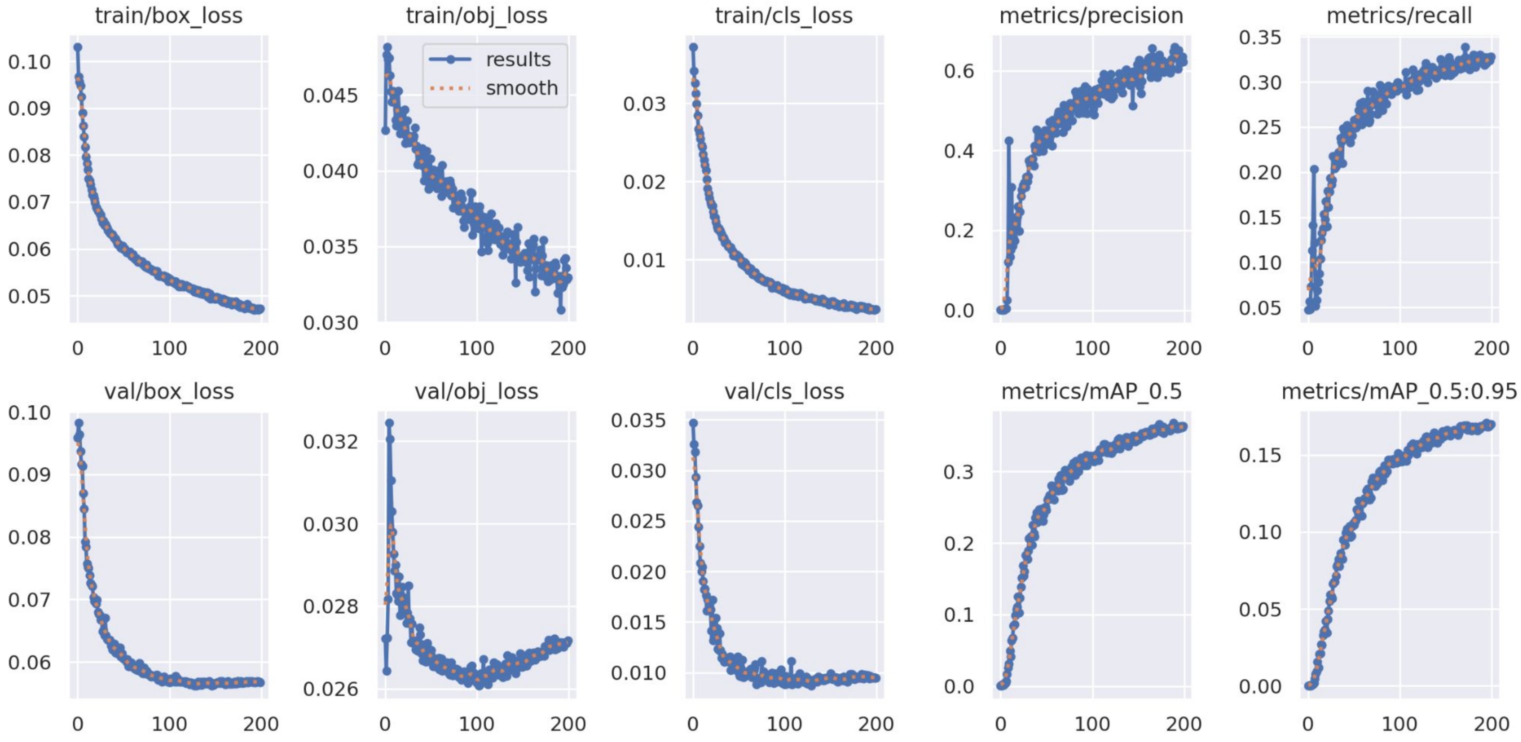
- Increase number of epochs (from 50 to 100 or 200)
- Adding dropout layers
- Increase network depth and width to 0.5

YOLOv5n' Architecture

- Learning rate: 0.01
- Loss functions:
  - **Classification loss:**
    - Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss)
    - FocalLoss (variant of cross-entropy that handles class imbalance) if gamma (focusing param) > 0
  - **Bounding Box regression loss:**
    - Complete Intersection over Union (IoU) loss
- Activation function
  - **SiLU** (Swish) Activation Function (default)
- Dropout regularization

```
# Parameters
nc: 3 # number of classes (modified)
depth_multiple: 0.5 # model depth multiple (modified)
width_multiple: 0.5 # layer channel multiple (modified)
anchors:
  - [10, 13, 16, 30, 33, 23] # P3/8
  - [30, 61, 62, 45, 59, 119] # P4/16
  - [116, 90, 156, 198, 373, 326] # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [
    [-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
    [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
    [-1, 3, C3, [128]],
    [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
    [-1, 6, C3, [256]],
    [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
    [-1, 9, C3, [512]],
    [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
    [-1, 3, C3, [1024]],
    [-1, 1, nn.Dropout, [0.2]],  # Adding Dropout with 20% probability
    [-1, 1, SPPF, [1024, 5]], # 9
  ]


 # YOLOv5 v6.0 head
 head: [
     [-1, 1, Conv, [512, 1, 1]],
     [-1, 1, nn.Upsample, [None, 2, "nearest"]],
     [[-1, 6], 1, Concat, [1]], # cat backbone P4
     [-1, 3, C3, [512, False]], # 13
     [-1, 1, nn.Dropout, [0.2]],  # Adding Dropout with 20% probability

     [-1, 1, Conv, [256, 1, 1]],
     [-1, 1, nn.Upsample, [None, 2, "nearest"]],
     [[-1, 4], 1, Concat, [1]], # cat backbone P3
     [-1, 3, C3, [256, False]], # 17 (P3/8-small)
     [-1, 1, nn.Dropout, [0.2]],  # Adding Dropout with 20% probability

     [-1, 1, Conv, [256, 3, 2]],
     [[-1, 14], 1, Concat, [1]], # cat head P4
     [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
     [-1, 1, nn.Dropout, [0.3]],  # Adding Dropout with 30% probability

     [-1, 1, Conv, [512, 3, 2]],
     [[-1, 10], 1, Concat, [1]], # cat head P5
     [-1, 3, C3, [1024, False]], # 23 (P5/32-large)
     [-1, 1, nn.Dropout, [0.5]],  # Adding Dropout with 50% probability

     [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
  ]
```

Evaluation metrics



Inference Results

Inference

In [33]:

```
!python detect.py --weights {wts} --img 640 --conf 0.25 --source data/images
```

```
detect: weights=['runs/train/sknav2/exp4/weights/best.pt'], source=data/images, data=data/coco128.yaml, imgsz=[64
0, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_csv=False, s
ave_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, u
pdate=False, project=runs/detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False,
half=False, dnn=False, vid_stride=1
YOLOv5 🚀 v7.0-327-g098ce03f Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
YOLOv5n_mod summary: 202 layers, 9069352 parameters, 0 gradients, 20.2 GFLOPs
image 1/2 /content/yolov5/data/images/bus.jpg: 640x480 2 persons, 49.9ms
image 2/2 /content/yolov5/data/images/zidane.jpg: 384x640 2 persons, 50.9ms
Speed: 0.5ms pre-process, 50.4ms inference, 249.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp3
```

Inference Results

Limitations/Potential Avenues for Improvements

- Train YOLOv5n’ on full COCO 2017 train data instead of only a subset
- Increase network depth and width
- Increase number of epochs to 300
- Customize data augmentations prior to training by modifying appropriate hyp.yaml configuration file: i.e., increase image size, shearing, rotate, flip, zoom in/out etc.
- Decrease learning rate from 0.01 to 0.001

Questions?

References/Resources

- https://github.com/tavjo/computer-vision-project/tree/main
- http://cocodataset.org/
- https://github.com/cocodataset/cocoapi/tree/master
- Jocher, G. (2020). YOLOv5 by Ultralytics (Version 7.0) [Computer software]. https://doi.org/10.5281/zenodo.3908559
- Deng, L., Bi, L., Li, H. *et al.* Lightweight aerial image object detection algorithm based on improved YOLOv5s. *Sci Rep* 13, 7817 (2023). https://doi.org/10.1038/s41598-023-34892-4