Laborprojekt Bildverarbeitung für Robotik 2023
Dirt Detection for Cleaning Robots

Group Members

Yutong Chen 3566907

Jiaming Yu 3532601

Hailin Chen 3498017

Why YOLO?

- **Speed**

  YOLOv4 can achieve a processing speed of over 60 with AP over 0.4 for COCO dataset.

- **Simplicity**

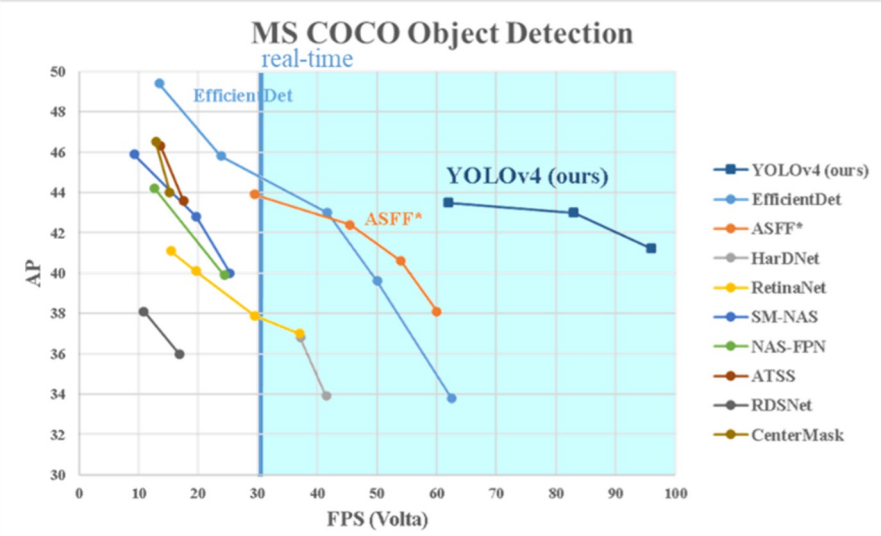
  Easy to understand and implement.

- **Global Context**

  Enabled to handle overlapping or adjacent objects more effectively

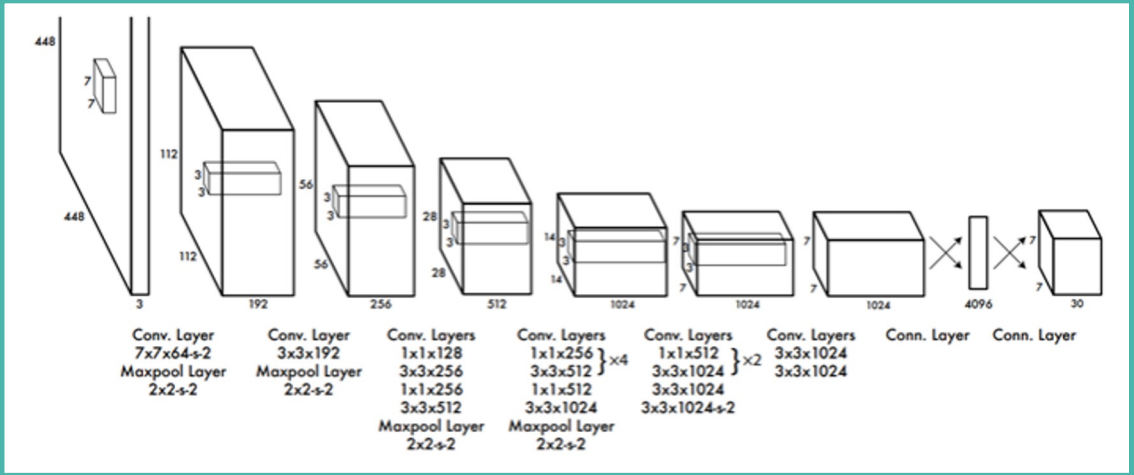
- **Object Relationships**

  Capable of capturing object relationships within an image

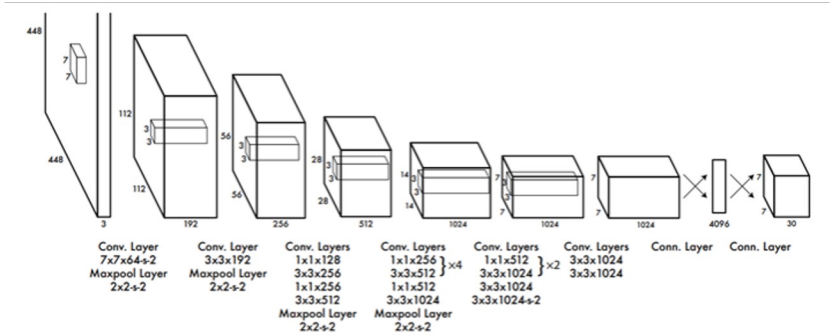


Performance of YOLO

Structure of YOLO



- 24 convolutional layers
- 2 fully connected layers
- An output layer of size 7×7×30



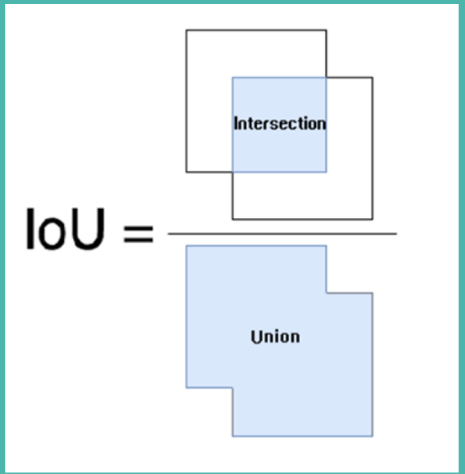
The output layer

- 7×7 grids, each gives out two predicted bounding boxes
- 4×2 parameters for descriptions of bounding boxes
- 2 parameters for confidence of objects

Confidence of result

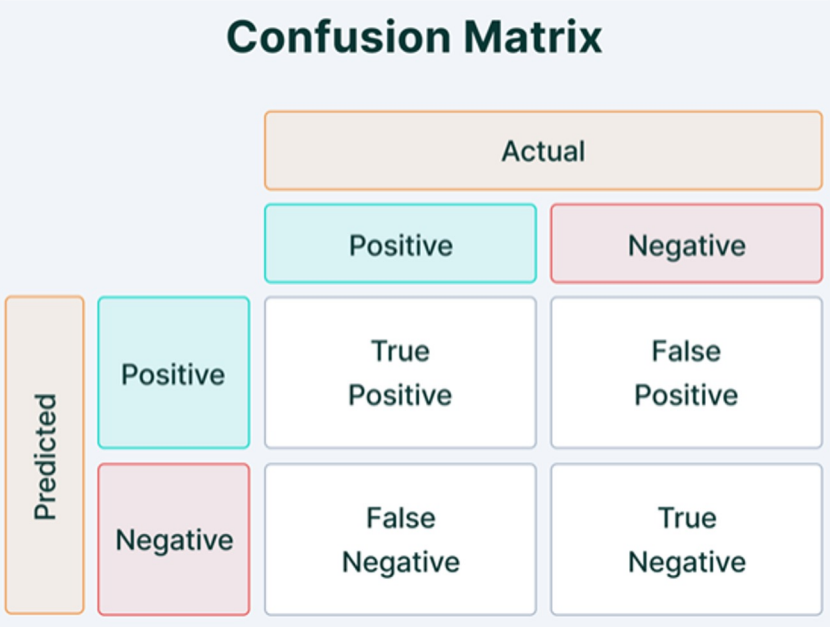
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

IoU: Intersection over Union



Performance indicators

- Precision
- Recall
- F1
- AP and mAP

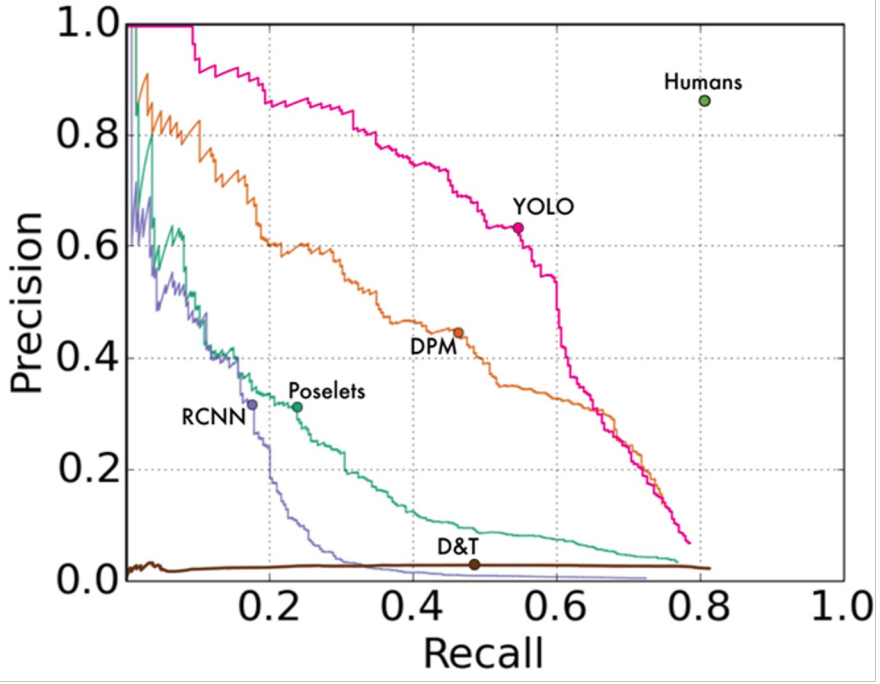


$$Precision = \frac{TP}{TP + FP}$$: How precise the search is.

$$Recall = \frac{TP}{TP + FN}$$: How complete the search is.

$$F_1\ score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}}$$: Harmonic mean of recall and precision

The P-R curve



- **AP: Average Precision at various recall levels.**

  Typically calculated by computing the area under the precision-recall curve.

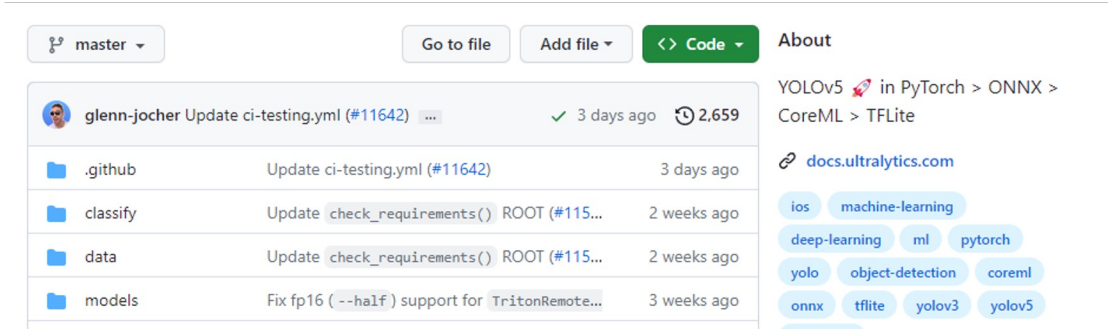
- **mAP: The average of AP across all classes.**

What have we done?

- Environment Configuration
- Transfer datasets into yolo format
- Training data
- First Look at the results
- Comparison between Yolos

Environment Configuration

- Python environment
- CUDA and cuDNN
- PyTorch
- Yolov5 code respository

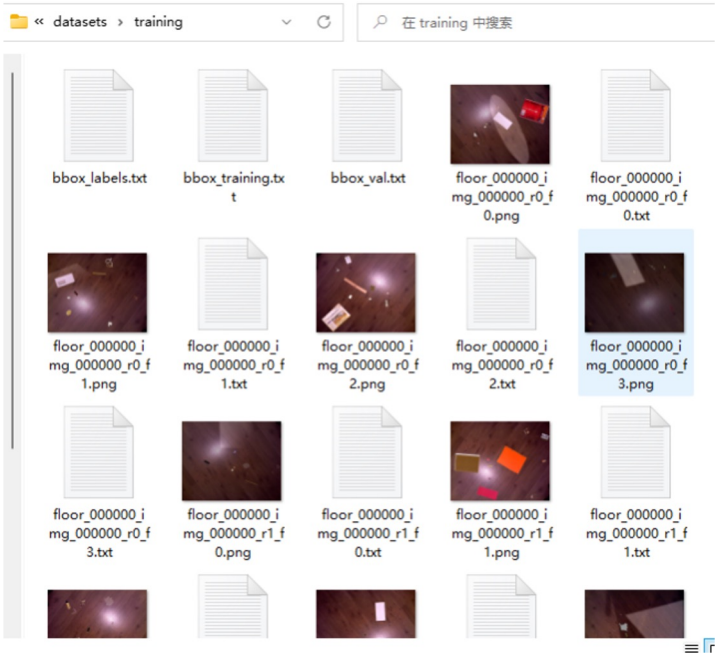


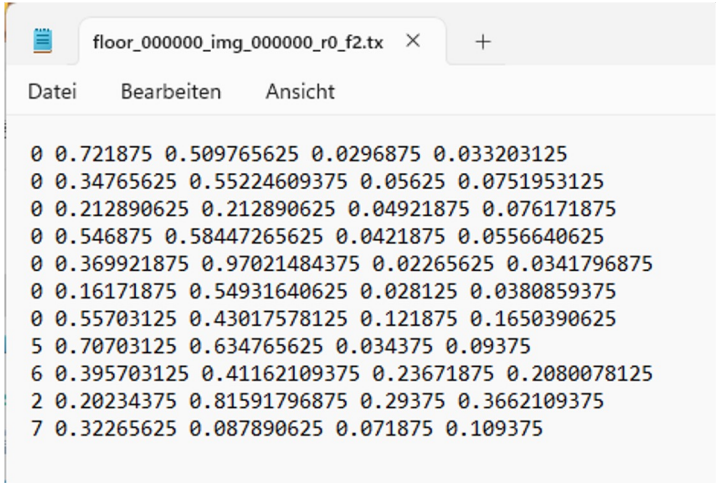
Transfer datasets into yolo format

```python
import os

mapping =  {
  'dirt': 0,
  'pens': 1,
  'paper_and_notebooks':2,
  'keys': 3,
  'usb_sticks': 4,
  'other': 5,
  'rulers': 6,
  'business_cards': 7,
  'scissors': 8,
  'tapes': 9}

def convert_to_yolo_format(l,image_height=1024,image_width=1280):

    class_id = mapping[l[5]]  # 对象类别
    x_min, y_min, x_max, y_max = map(float, l[1:5])  # 边界框坐标

        # 计算边界框中心坐标、宽度和高度，并将其归一化
    x_center = (x_min + x_max) / (2.0 * image_width)
    y_center = (y_min + y_max) / (2.0 * image_height)
    bbox_width = (x_max - x_min) / image_width
    bbox_height = (y_max - y_min) / image_height

    # 将转换后的标签信息添加到列表中
    yolo_label = f"{class_id} {x_center} {y_center} {bbox_width} {bbox_height}"

    return yolo_label


file_path = r"C:\Users\c1257\Desktop\yolov5-master\datasets\test\blended_training_floor_images (TestSynthTF)"
with open(os.path.join(file_path,'bbox_labels.txt'), "r") as file:
    # 逐行读取文件内容
    line = file.readline()
    l = line.split()
    pic_name = l[0]
    label_file = open(os.path.join(file_path, pic_name+".txt"),"w")
    output = convert_to_yolo_format(l)
    label_file.write(output+"\n")
    line = file.readline()
```


```python
    while line:
        # 处理每行的数据
        l = line.split()
        if l[0] == pic_name:
            output = convert_to_yolo_format(l)
            label_file.write(output+"\n")
        else:
            label_file.close()
            l = line.split()
            pic_name = l[0]
            label_file = open(os.path.join(file_path, pic_name+".txt"),"w")
            output = convert_to_yolo_format(l)
            label_file.write(output+"\n")


        # 读取下一行
        line = file.readline()
```

Transfer datasets into yolo format





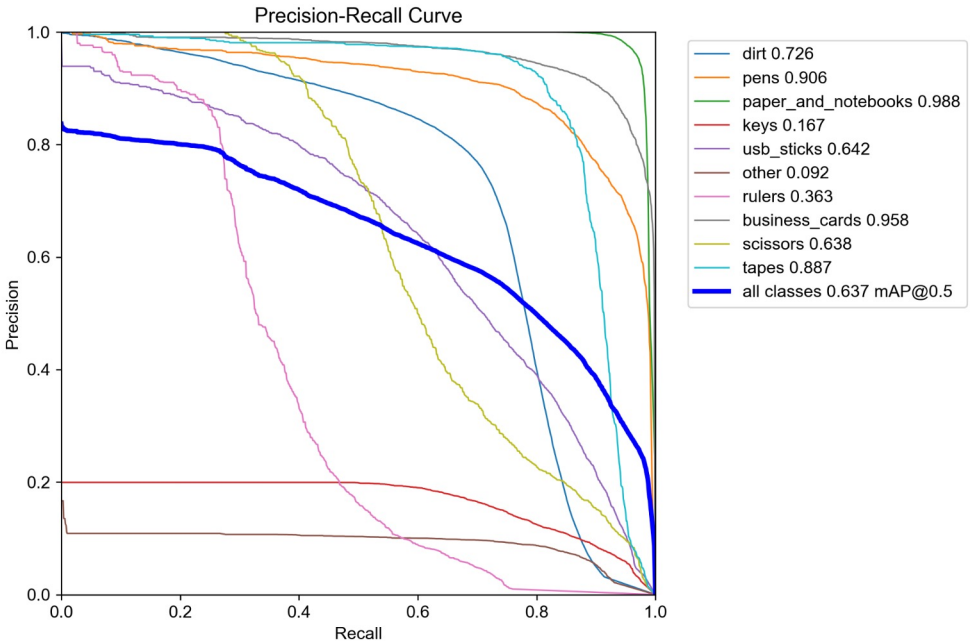
Training Data

```yaml
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt,
path: ../datasets
train: # train images (relative to 'path')  9248 images
  - training
val: # val images (relative to 'path')  4952 images
  - training
test: # test images (optional)
  - test/blended_training_floor_images (TestSynthTF)

# Classes
names:
  0: dirt
  1: pens
  2: paper_and_notebooks
  3: keys
  4: usb_sticks
  5: other
  6: rulers
  7: business_cards
  8: scissors
  9: tapes
```

- create a yaml file
- implement weight files (yolov5s.pt)
- python train.py --img 640 --epochs 3 --data voc_dirt.yaml --weights yolov5s.pt

Results from training data

Results from training data



Training dataset of 9248 samples

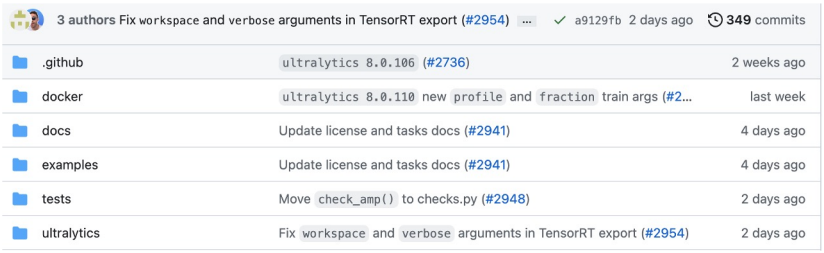
YOLOv8 Implementation

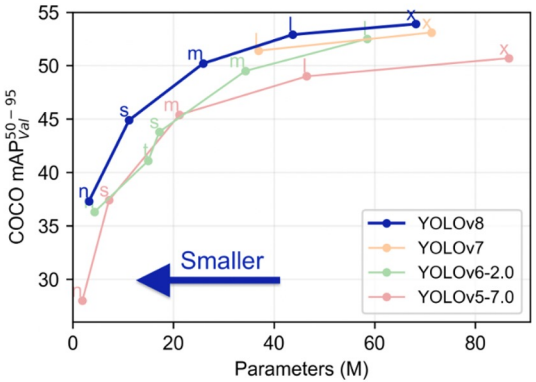
- Comparing YOLOv8 and YOLOv5
- Results Improvement (Next Steps)
- Data Augmentation
- Problems

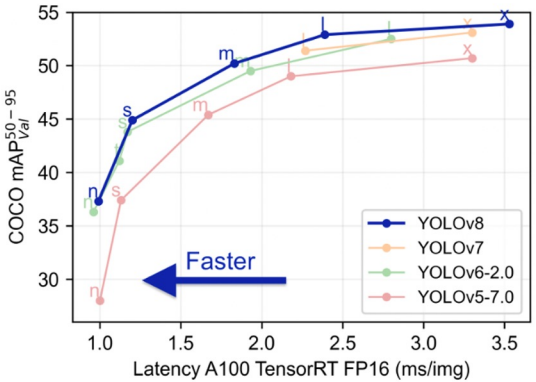
YOLOv8 vs YOLOv5

Differences:

- easy-to-use CLI
- well-structured Python package
- Accuracy Improvements
- Limited speed difference







Object Detection Performance Comparison (YOLOv8 vs YOLOv5)

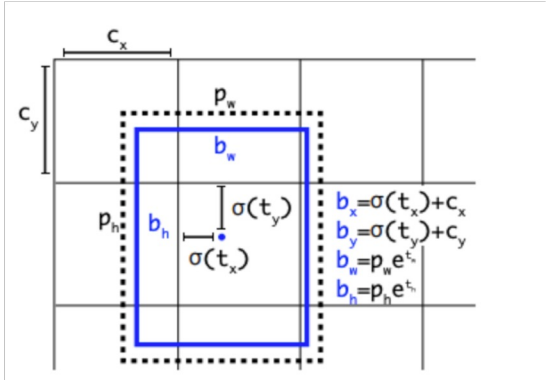
Model Size	YOLOv5	YOLOv8	Difference
Nano	28	37.3	+33.21%
Small	37.4	44.9	+20.05%
Medium	45.4	50.2	+10.57%
Large	49	52.9	+7.96%
Xtra Large	50.7	53.9	+6.31%

*Image Size = 640

YOLOv8 vs YOLOv5

Similarities:

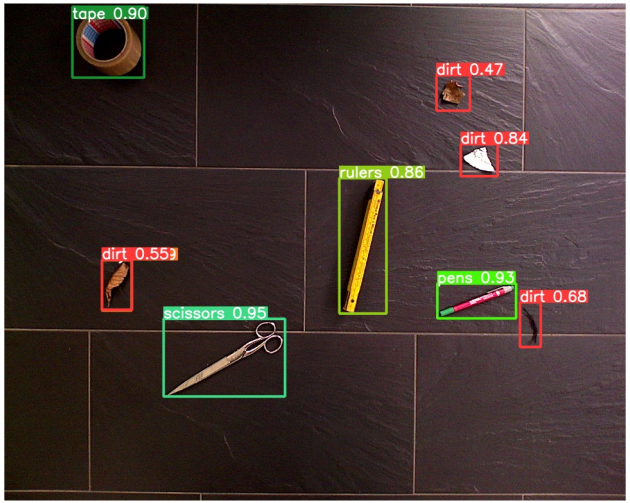
- Anchor Boxes
- Non-Maximum Suppression
- Optimizer(Adam optimizer)
- Activation Function(Mish activation function)



Visualization of an anchor box in YOLO

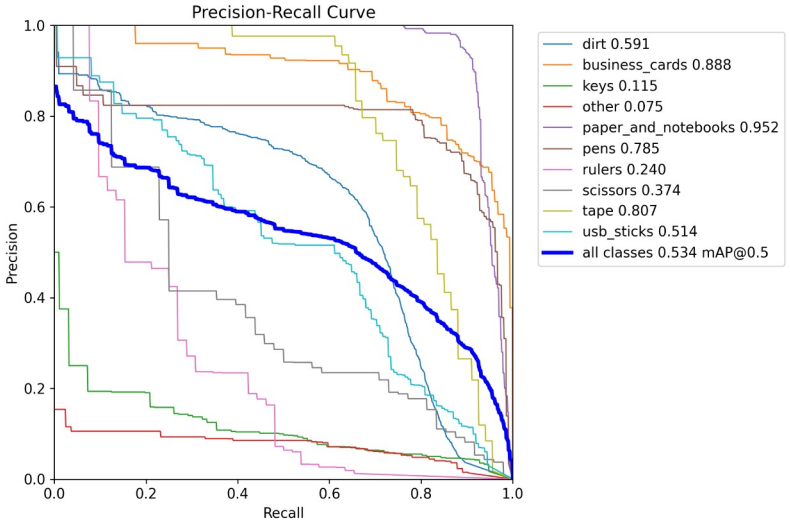
floor1.txt

```
0 0.278125 0.80029296875 0.0390625 0.0498046875
0 0.509765625 0.9150390625 0.05703125 0.091796875
0 0.821484375 0.67138671875 0.11640625 0.1044921875
0 0.804296875 0.35400390625 0.05078125 0.0556640625
0 0.861328125 0.3955078125 0.06953125 0.0390625
0 0.8484375 0.3310546875 0.0421875 0.048828125
6 0.483984375 0.232421875 0.26484375 0.1640625
7 0.690234375 0.1904296875 0.08046875 0.26953125
4 0.329296875 0.4619140625 0.21171875 0.2734375
```

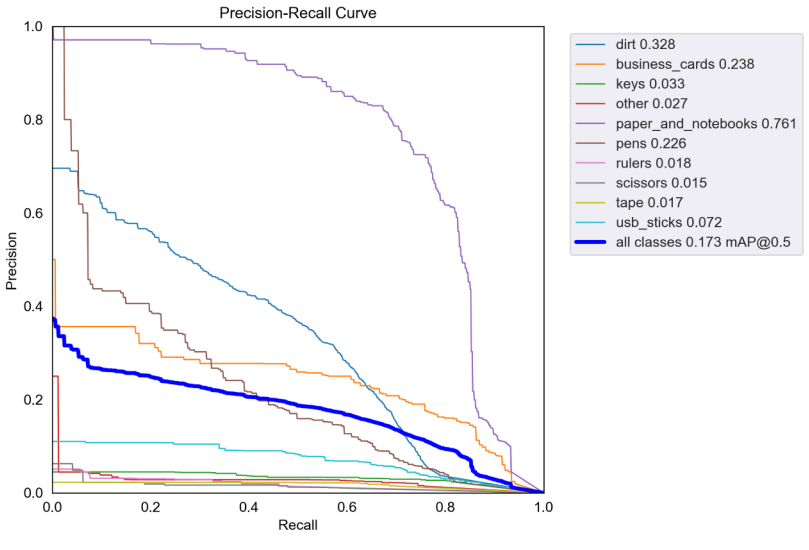


YOLOv8 vs YOLOv5

Results:



YOLOv8n(3 Epochs, 8 Batches)



YOLOv5n(3 Epochs, 8 Batches)

Results Improvement

Next Steps:

- Trying different model type (YOLOv8s)
- Training with larger dataset (≥ 1500 images per class recommended)
- Training Settings (more Epochs without overfitting and larger Batch size)
- Data augmentation (Hyperparameters modifying in default.yaml file)

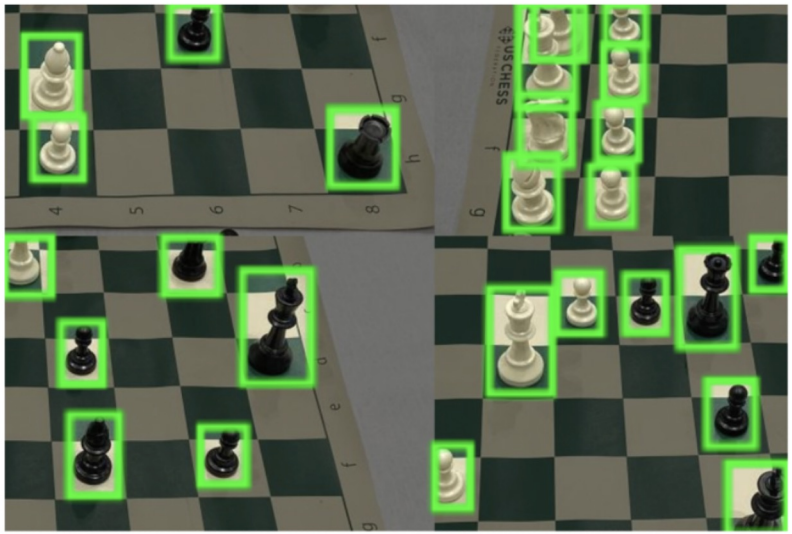
Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

```
# Hyperparameters ------------------------------------------------------------------------------------------
lr0: 0.01  # initial learning rate (i.e. SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final 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: 7.5  # box loss gain
cls: 0.5  # cls loss gain (scale with pixels)
dfl: 1.5  # dfl loss gain
pose: 12.0  # pose loss gain
kobj: 1.0  # keypoint obj loss gain
label_smoothing: 0.0  # label smoothing (fraction)
nbs: 64  # nominal batch size
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)
```

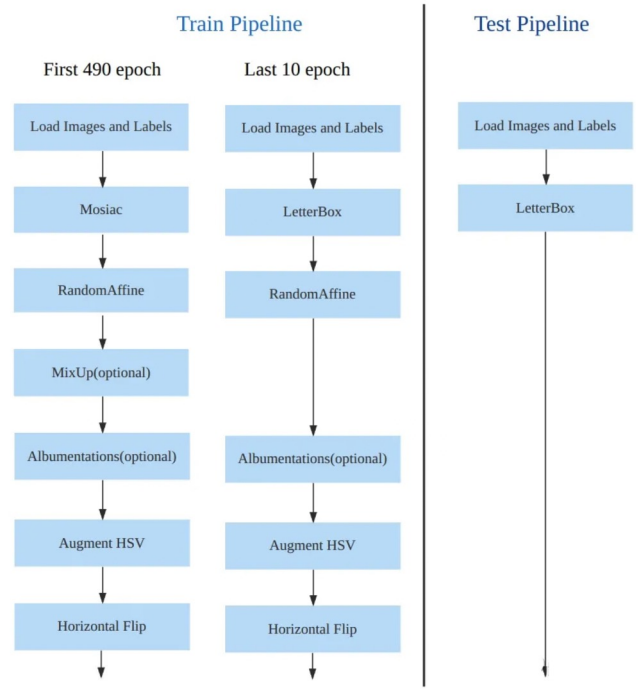
Data Augmentation

Mosaic Augmentation:

- Take 4 images and combine them into a single image as the new mosaic image.



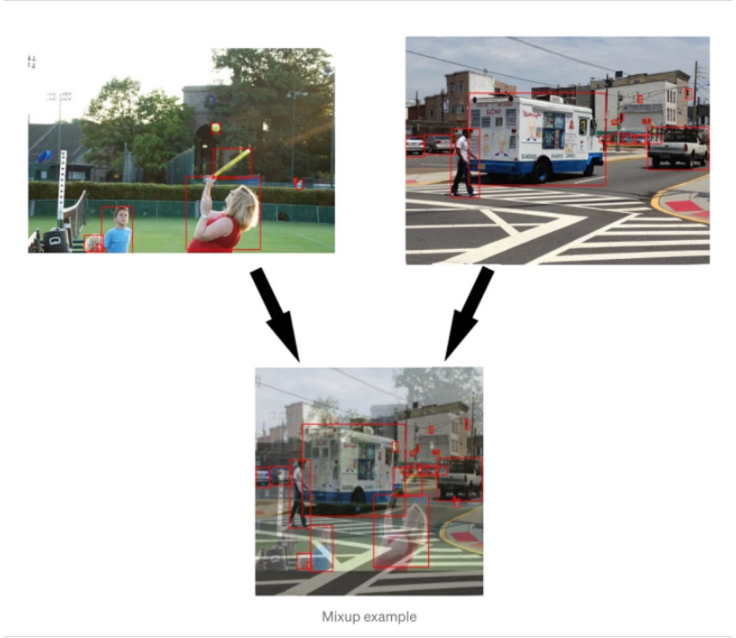
Mosaic augmentation of chess board photos



Data Augmentation

Mixup Augmentation:

- Combine two different images and their associated labels to create a new synthetic sample.



Mixup example

Data Augmentation

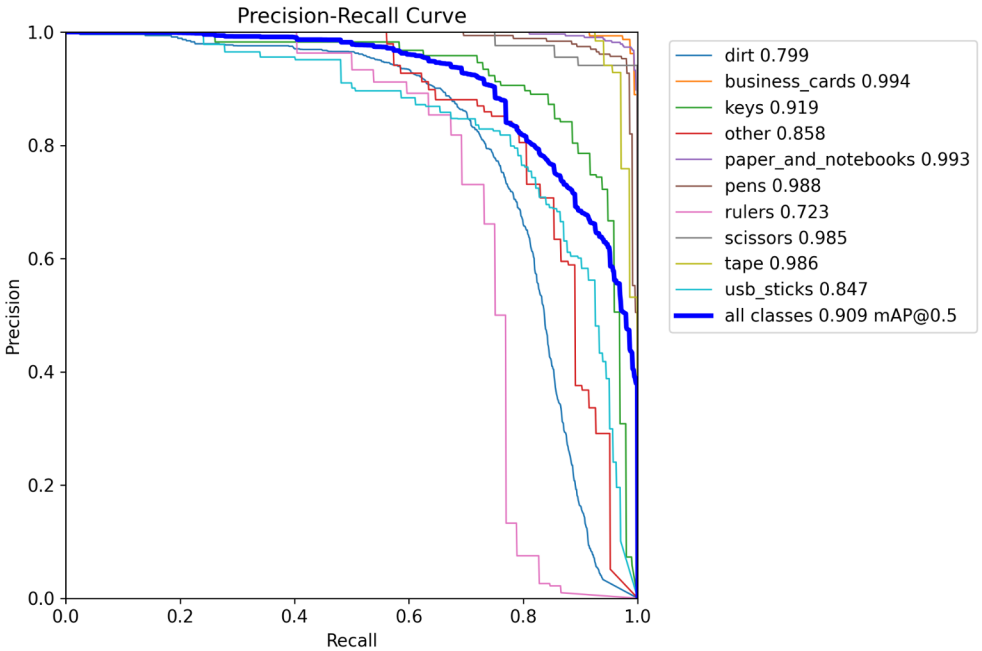
Label Smoothing:

- This involves modifying the ground truth labels during training to introduce a small amount of uncertainty or noise in the label assignments.

Predicted prob (p) :	0.3	0.6	0.01	0.01	0.05	0.03
Smoothed label :	0.05	0.75	0.05	0.05	0.05	0.05
Label one-hot (y) :	0	1	0	0	0	0
Label name :	飞机	鸟	猫	狗	汽车	拖拉机

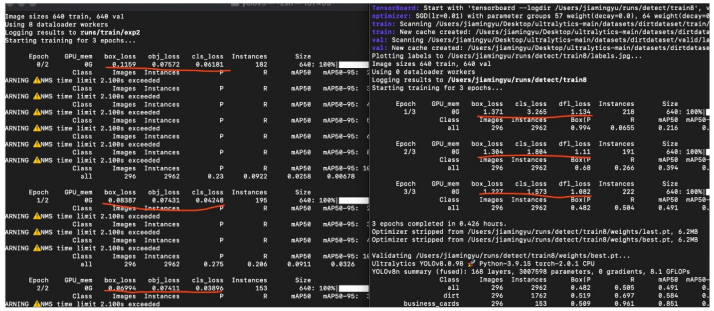
Latest Result

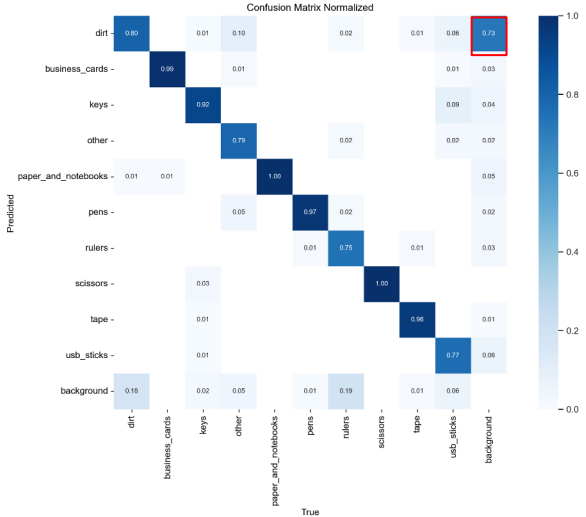
YOLOv8s model (30 epochs, 8batches, 1156 pictures for training and validation, 80 pictures for test)

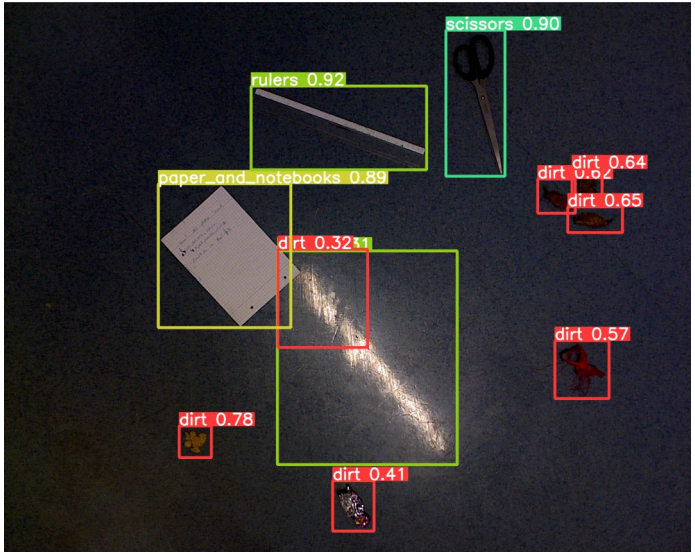


Problems:

- Why does YOLOv8 have larger loss but still higher mAP scores than YOLOv5?
- How to avoid the background being detected as dirt.







Thank you for your attention!