

# **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.

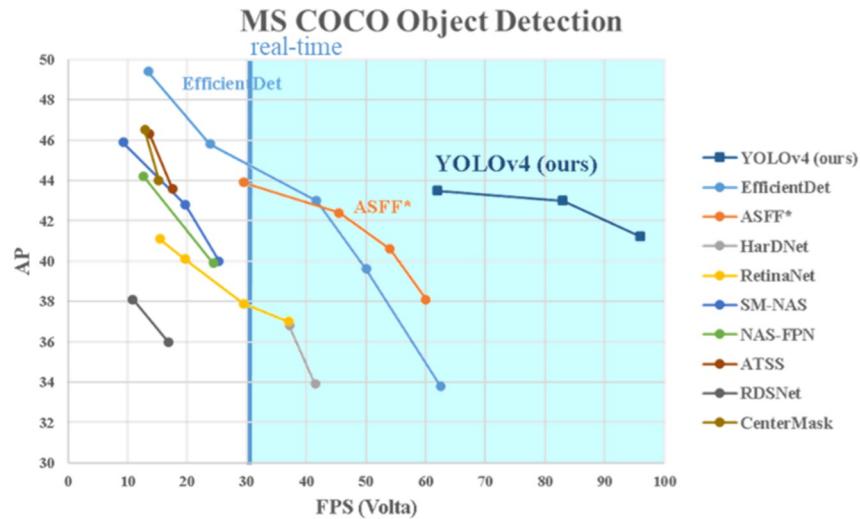
- **Global Context**

Enabled to handle overlapping or adjacent objects more effectively

- **Object Relationships**

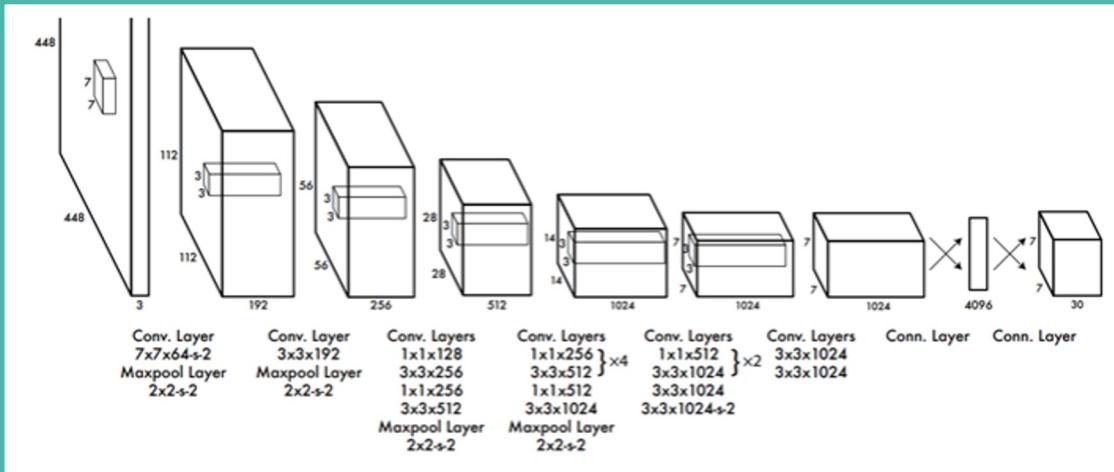
Capable of capturing object relationships within an image

---



# Performance of YOLO

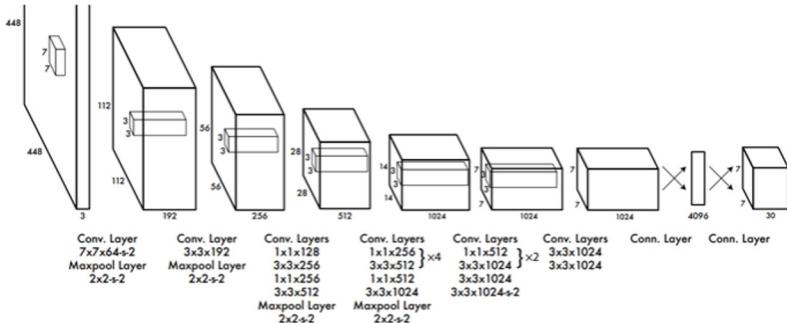
# Structure of YOLO



- **24 convolutional layers**
- **2 fully connected layers**
- **An output layer of size  $7 \times 7 \times 30$**

# The output layer

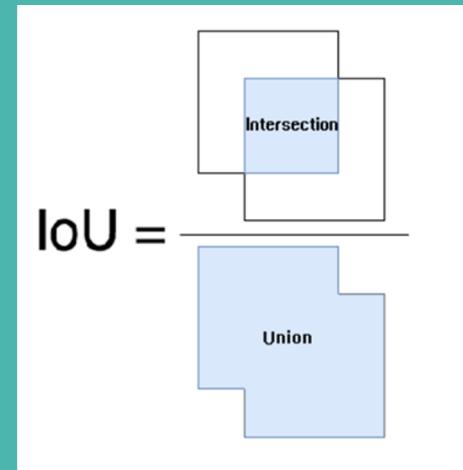
- $7 \times 7$  grids, each gives out two predicted bounding boxes
- $4 \times 2$  parameters for descriptions of bounding boxes
- 2 parameters for confidence of objects



# 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
-

## Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

---

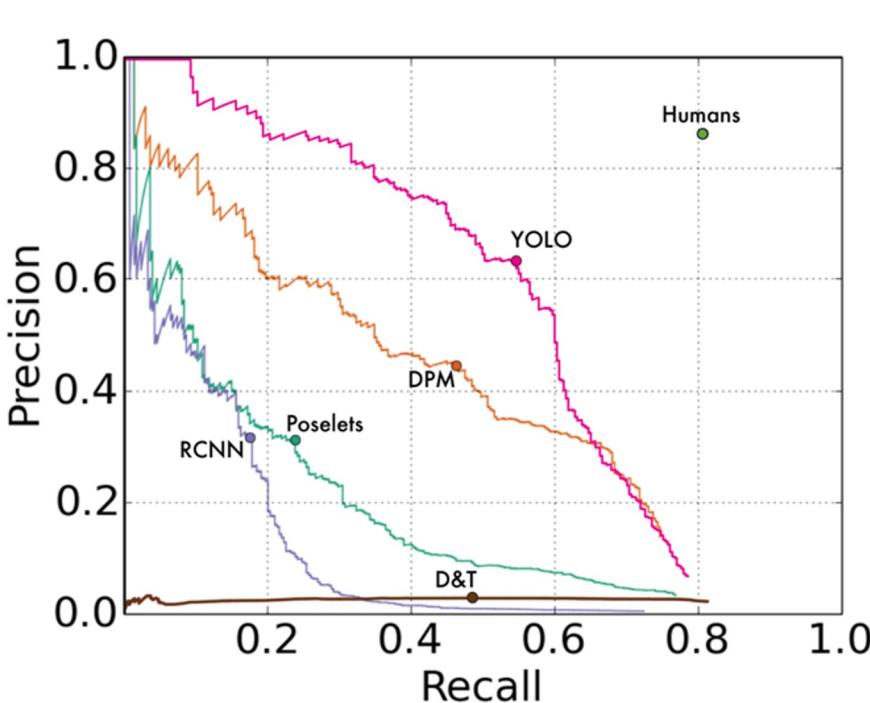
$Precision = \frac{TP}{TP + FP}$ : How precise the search is.

$Recall = \frac{TP}{TP + FN}$ : 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 repository



A screenshot of a GitHub repository page for 'YOLOv5' in PyTorch. The repository has 2,659 stars and was last updated 3 days ago. The page shows several commits from 'glenn-jocher' and other contributors, including updates to CI testing files and requirements. The right sidebar includes links to documentation at 'docs.ultralytics.com' and various technology tags: ios, machine-learning, deep-learning, ml, pytorch, yolo, object-detection, coreml, onnx, tflite, yolov3, and yolov5.

Commit	Message	Time Ago
glenn-jocher Update ci-testing.yml (#11642) ...	Update ci-testing.yml (#11642) ...	3 days ago
.github	Update ci-testing.yml (#11642)	3 days ago
classify	Update check_requirements() ROOT (#115...)	2 weeks ago
data	Update check_requirements() ROOT (#115...)	2 weeks ago
models	Fix fp16 ( --half ) support for TritonRemote...	3 weeks ago

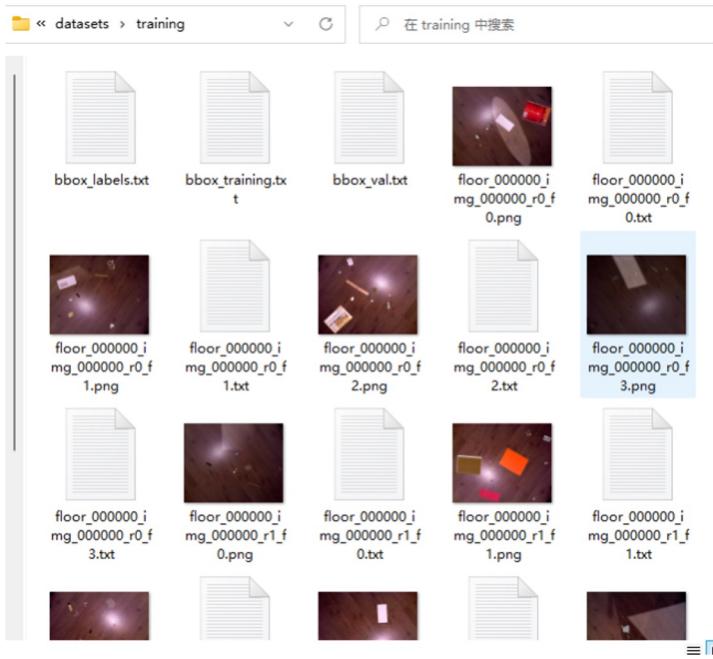
# Transfer datasets into yolo format

```
1 import os
2
3 mapping = {
4     'dirt': 0,
5     'pens': 1,
6     'paper_and_notebooks': 2,
7     'keys': 3,
8     'usb_sticks': 4,
9     'other': 5,
10    'rulers': 6,
11    'business_cards': 7,
12    'scissors': 8,
13    'tapes': 9}
14
15 def convert_to_yolo_format(l,image_height=1024,image_width=1280):
16
17     class_id = mapping[l[5]] # 对象类别
18     x_min, y_min, x_max, y_max = map(float, l[1:5]) # 边界框坐标
19
20     # 计算边界框中心坐标、宽度和高度，并将其归一化
21     x_center = (x_min + x_max) / (2.0 * image_width)
22     y_center = (y_min + y_max) / (2.0 * image_height)
23     bbox_width = (x_max - x_min) / image_width
24     bbox_height = (y_max - y_min) / image_height
25
26     # 将转换后的标签信息添加到列表中
27     yolo_label = f'{class_id} {x_center} {y_center} {bbox_width} {bbox_height}'
28
29     return yolo_label
30
31
32 file_path = r'C:\Users\c1257\Desktop\yolov5-master\datasets\test\blended_training_floor_images (TestSynthTF)'
33 with open(os.path.join(file_path,'bbox_labels.txt'), "r") as file:
34     # 逐行读取文件内容
35     line = file.readline()
36     l = line.split()
37     pic_name = l[0]
38     label_file = open(os.path.join(file_path, pic_name+".txt"), "w")
39     output = convert_to_yolo_format(l)
40     label_file.write(output+"\n")
41     line = file.readline()
```



```
42     while line:
43         # 处理每行的数据
44         l = line.split()
45         if l[0] == pic_name:
46             output = convert_to_yolo_format(l)
47             label_file.write(output+"\n")
48         else:
49             label_file.close()
50             l = line.split()
51             pic_name = l[0]
52             label_file = open(os.path.join(file_path, pic_name+".txt"), "w")
53             output = convert_to_yolo_format(l)
54             label_file.write(output+"\n")
55
56
57         # 读取下一行
58         line = file.readline()
```

# Transfer datasets into yolo format



A text editor window showing a single file named 'floor\_000000\_img\_000000\_r0\_f2.txt'. The file contains the following YOLO formatted data:

```
0 0.721875 0.509765625 0.0296875 0.033203125
0 0.34765625 0.55224609375 0.05625 0.0751953125
0 0.212890625 0.212890625 0.04921875 0.076171875
0 0.546875 0.58447265625 0.0421875 0.0556640625
0 0.369921875 0.97021484375 0.02265625 0.0341796875
0 0.16171875 0.54931640625 0.028125 0.0380859375
0 0.55703125 0.43017578125 0.121875 0.1650390625
5 0.70703125 0.634765625 0.034375 0.09375
6 0.395703125 0.41162109375 0.23671875 0.2080078125
2 0.20234375 0.81591796875 0.29375 0.3662109375
7 0.32265625 0.087890625 0.071875 0.109375
```

# Training Data

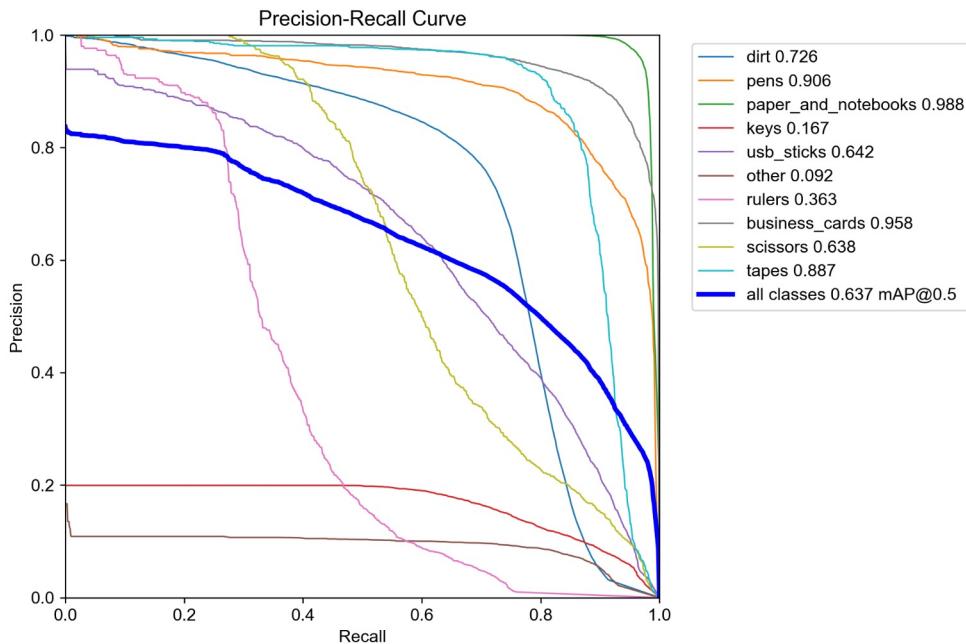
- create a yaml file
- implement weight files (yolov5s.pt)
- python train.py --img 640 --epochs 3 --data  
voc\_dirt.yaml --weights yolov5s.pt

```
10 # Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt,
11 path: ../../datasets
12 train: # train images (relative to 'path') 9248 images
13 | - training
14 val: # val images (relative to 'path') 4952 images
15 | - training
16 test: # test images (optional)
17 | - test/blended_training_floor_images (TestSynthTF)
18
19 # Classes
20 names:
21 | 0: dirt
22 | 1: pens
23 | 2: paper_and_notebooks
24 | 3: keys
25 | 4: usb_sticks
26 | 5: other
27 | 6: rulers
28 | 7: business_cards
29 | 8: scissors
30 | 9: tapes
```

# Results from training data



# Results from training data



Training dataset of 9248 samples

# YOLOv8 Implementation

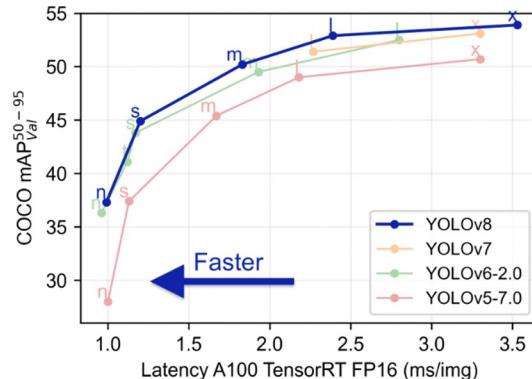
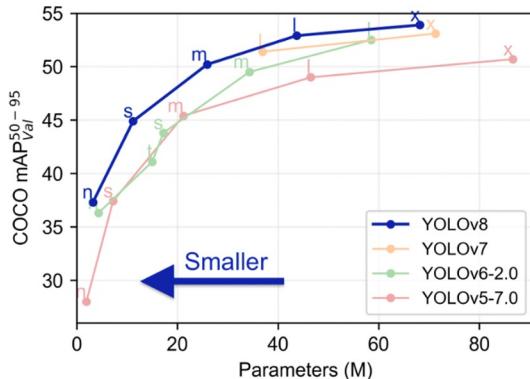
- Comparing YOLOv8 and YOLOv5
- Results Improvement (Next Steps)
- Data Augmentation
- Problems

# YOLOv8 vs YOLOv5

## Differences:

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

 3 authors Fix workspace and verbose arguments in TensorRT export (#2954) ... ✓ a9129fb 2 days ago	349 commits
.github	ultralytics 8.0.106 (#2736)
docker	ultralytics 8.0.110 new profile and fraction train args (#2...
docs	Update license and tasks docs (#2941)
examples	Update license and tasks docs (#2941)
tests	Move check_amp() to checks.py (#2948)
ultralytics	Fix workspace and verbose arguments in TensorRT export (#2954)



### 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%

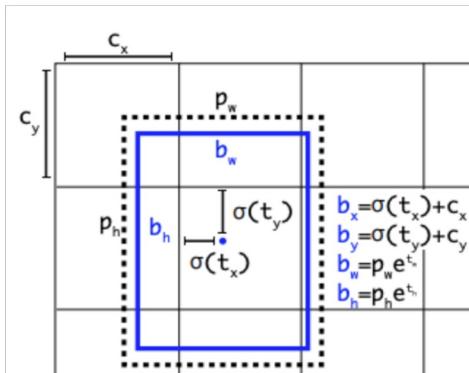
\*Image Size = 640

# YOLOv8 vs YOLOv5

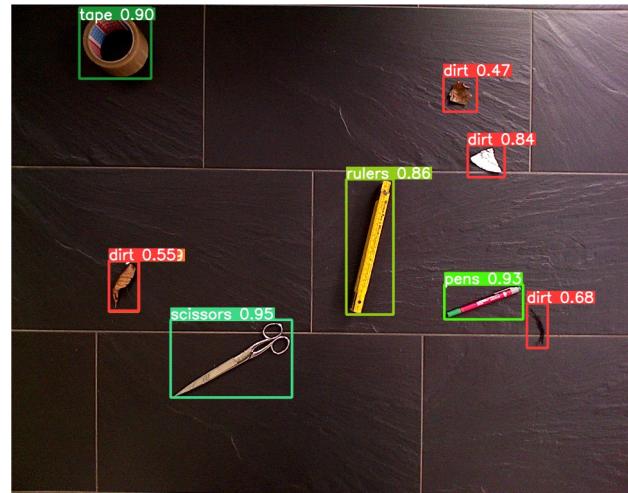
## Similarities:

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

```
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
```

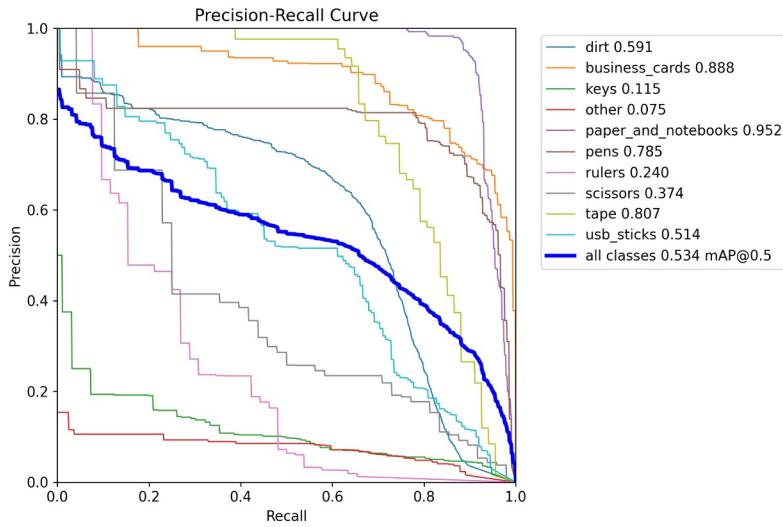


Visualization of an anchor box in YOLO

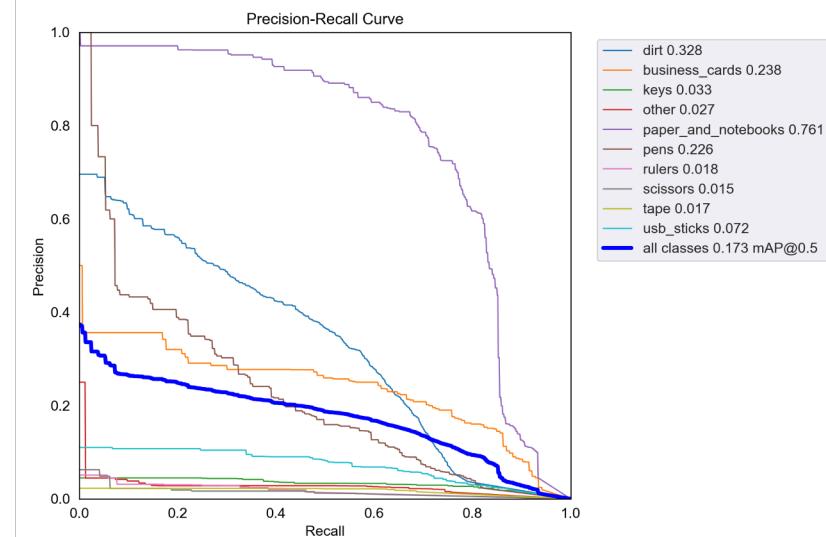


# YOLOv8 vs YOLOv5

## Results:



YOLOv8n(3 Epochs, 8 Batches)



YOLOv5n(3 Epochs, 8 Batches)

# Results Improvement

## Next Steps:

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

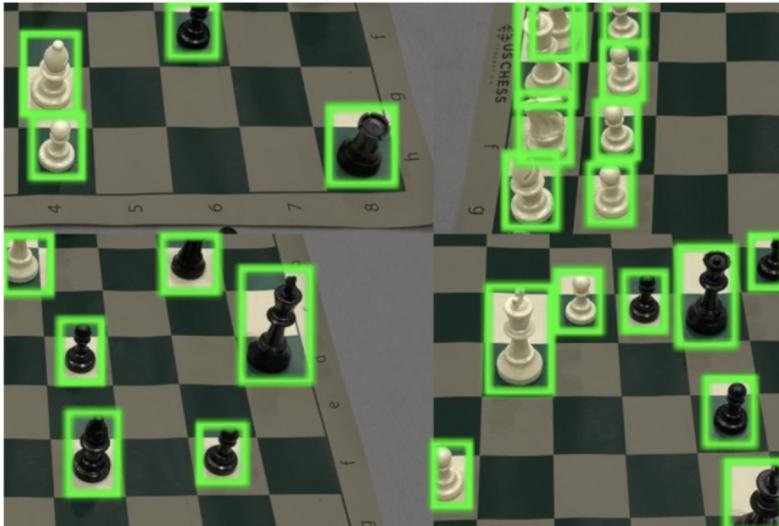
Model	size (pixels)	mAP <sub>val</sub> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

```
81 # Hyperparameters --  
82 lr0: 0.01 # initial learning rate (i.e. SGD=1E-2, Adam=1E-3)  
83 lrf: 0.01 # final learning rate (lr0 * lrf)  
84 momentum: 0.937 # SGD momentum/Adam beta1  
85 weight_decay: 0.0005 # optimizer weight decay 5e-4  
86 warmup_epochs: 3.0 # warmup epochs (fractions ok)  
87 warmup_momentum: 0.8 # warmup initial momentum  
88 warmup_bias_lr: 0.1 # warmup initial bias lr  
89 box: 7.5 # box loss gain  
90 cls: 0.5 # cls loss gain (scale with pixels)  
91 dfl: 1.5 # dfl loss gain  
92 pose: 12.0 # pose loss gain  
93 kobj: 1.0 # keypoint obj loss gain  
94 label_smoothing: 0.0 # label smoothing (fraction)  
95 nbs: 64 # nominal batch size  
96 hsv_h: 0.015 # image HSV-Hue augmentation (fraction)  
97 hsv_s: 0.7 # image HSV-Saturation augmentation (fraction)  
98 hsv_v: 0.4 # image HSV-Value augmentation (fraction)  
99 degrees: 0.0 # image rotation (+/- deg)  
100 translate: 0.1 # image translation (+/- fraction)  
101 scale: 0.5 # image scale (+/- gain)  
102 shear: 0.0 # image shear (+/- deg)  
103 perspective: 0.0 # image perspective (+/- fraction), range 0-0.001  
104 flipud: 0.0 # image flip up-down (probability)  
105 filp_lr: 0.5 # image flip left-right (probability)  
106 mosaic: 1.0 # image mosaic (probability)  
107 mixup: 0.0 # image mixup (probability)  
108 copy_paste: 0.0 # segment copy-paste (probability)
```

# Data Augmentation

## Mosaic Augmentation:

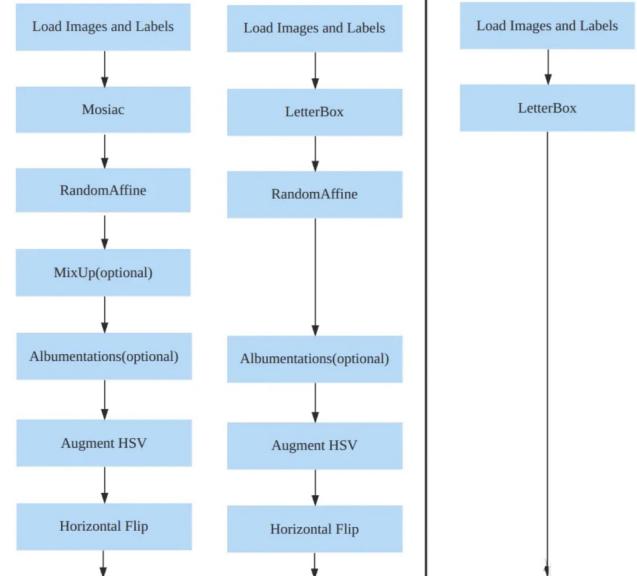
- Take 4 images and combine them into a single image as the new mosaic image.



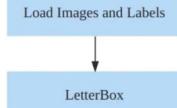
Mosaic augmentation of chess board photos

### Train Pipeline

First 490 epoch      Last 10 epoch



### Test Pipeline



# 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

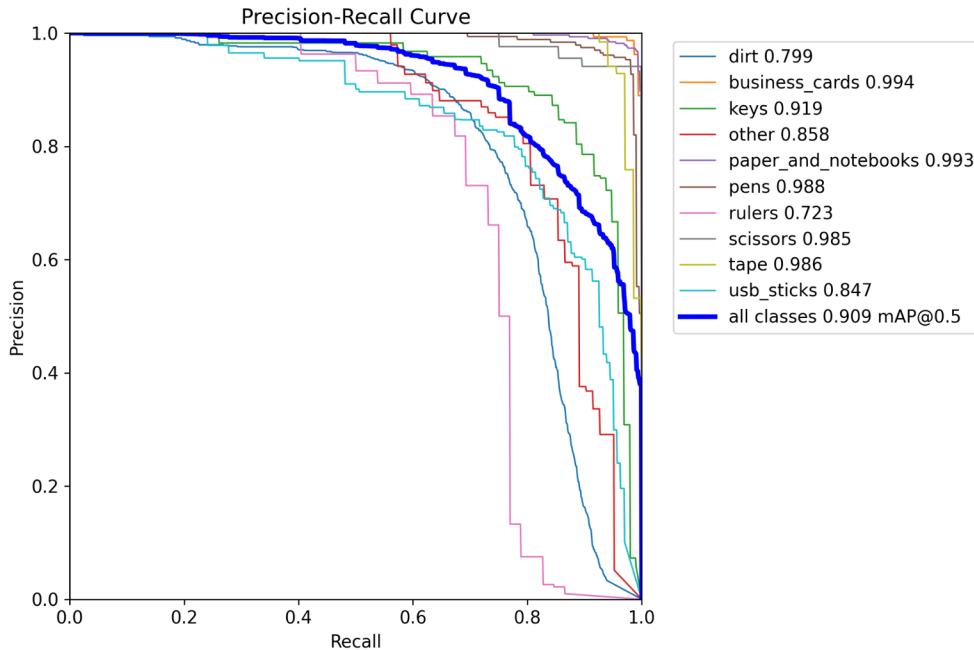
Smoothed label : 0.05 0.75 0.05 0.05 0.05 0.05

Label one-hot (y) : 0 1 0 0 0 0

Label name : 飞机 鸟 猫 狗 汽车 拖拉机

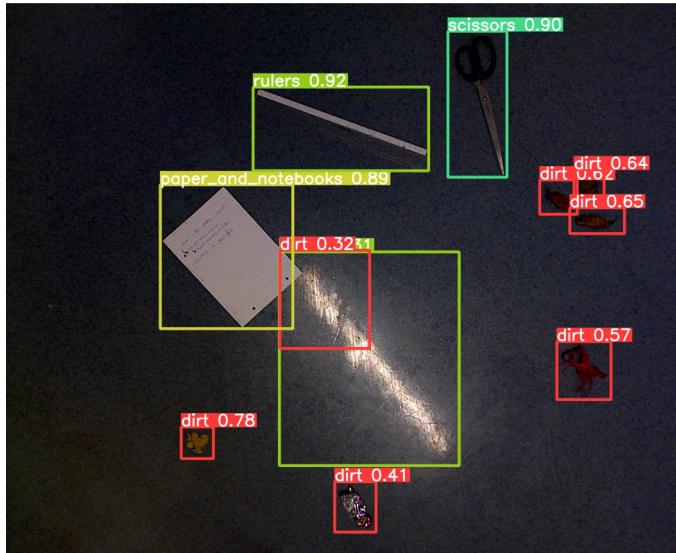
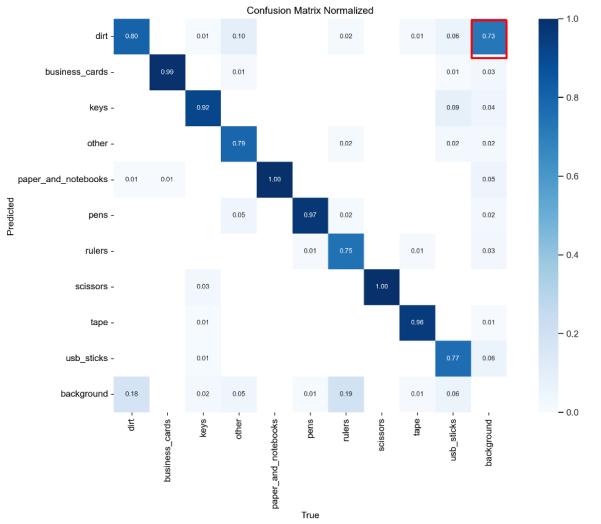
# Latest Result

YOLOv8s model (30 epochs, 8 batches, 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.



```
TensorBoard: Start with 'tensorboard --logdir /Users/jiamingyu/runs/detect/train'...
optimizers: SGD(1e-08) with parameter groups: 57 weight_decay=0.0, 64 weight_decay=0.001
Optimizer: SGD(1e-08)
Epoch: 0/3 GPU: 00 box_loss obj_loss cls_loss Instances Size
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Using 8 out of 8 workers
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Starting training for 3 epochs...
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Epoch: 1/3 GPU: 00 box_loss obj_loss cls_loss dfl_loss Instances Size
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
all 296 296 1.371 1.260 1.158 210 648 100% [0.8655]
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Epoch: 2/3 GPU: 00 box_loss obj_loss cls_loss dfl_loss Instances Size
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
all 296 296 0.994 0.894 0.8855 0.216 0.
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Epoch: 3/3 GPU: 00 box_loss obj_loss cls_loss dfl_loss Instances Size
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
all 296 296 0.68 0.68 0.68 222 648 100% [0.8746]
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Epoch: 0/3 GPU: 00 box_loss obj_loss cls_loss Instances Size
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
3 epochs completed in 0.426 hours.
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Optimizer stripped from /Users/jiamingyu/runs/detect/train/weights/best.pt.. 6 MB
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances P R mAP@0 mAP@0.5
Validating /Users/jiamingyu/runs/detect/train/weights/best.pt...
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances BoxP R mAP@0 mAP@0.5
Business cards 296 296 0.826 0.826 0.826 153 648 100% [0.851]
WARNING ▲ HWS time limit 2.08ms exceeded Class Images Instances BoxP R mAP@0 mAP@0.5
Business cards 296 296 0.509 0.509 0.509 153 648 100% [0.851]
```

# Thank you for your attention!