Laborprojekt Servicerobotik 2023
Abschlusspräsentation
Projekt 05: Dirt Detection for Cleaning Robots

Bearbeiter: Yutong Chen 3566907
Jiaming Yu 3532601
Hailin Chen 3498017

Motivation

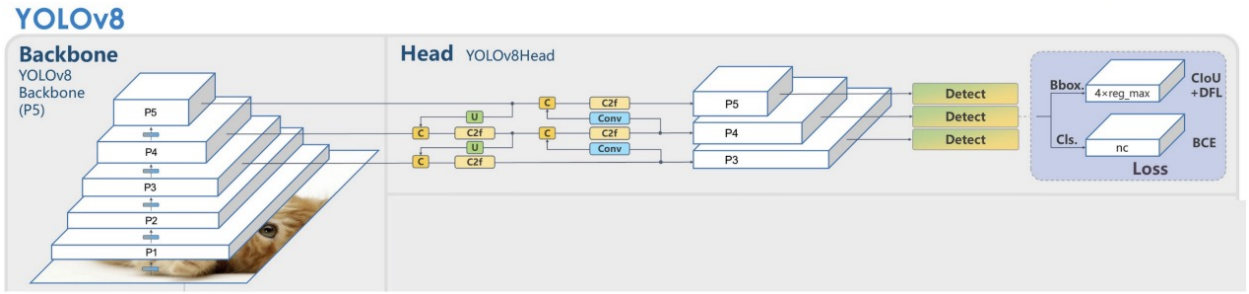
- Theme: Dirt Detection for Cleaning Robots
  - Distinguish dirt from other common office objects.
  - Class: **Dirt** (major), keys, paper and notebooks, etc. (common office objects)
  - Requirements:
    - Fast (real time detection)
    - Precise (small objects)
    - (Relatively) Low computational demands.

Concept

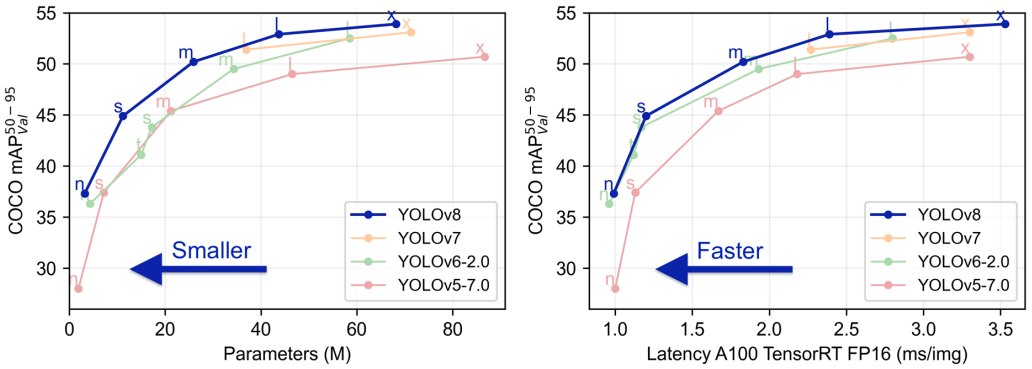
- Content
  - Model
  - Dataset transformation
  - YOLOv8 implementation
  - Data augmentation
  - Hyperparameter tuning
  - Result
  - Summary

Model

- YOLOv8
  - Architecture[1]:
    - Path Aggregation Network (PAN): suitable for objects in various scale.

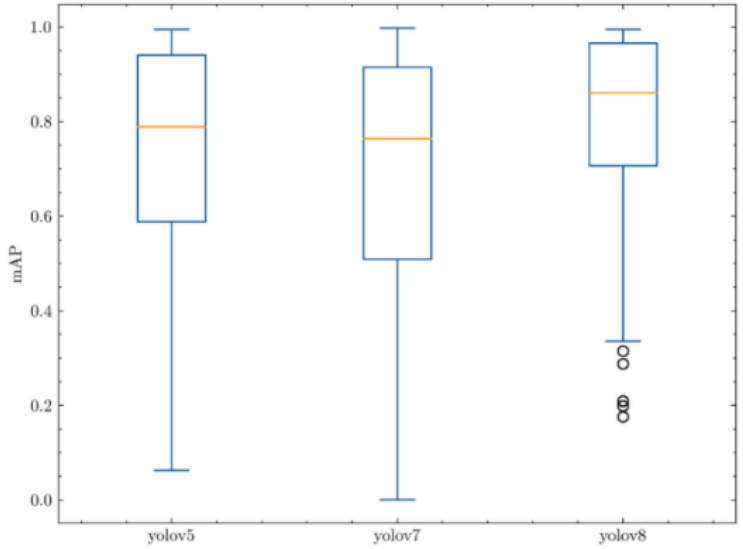


  - Model parameters[2]

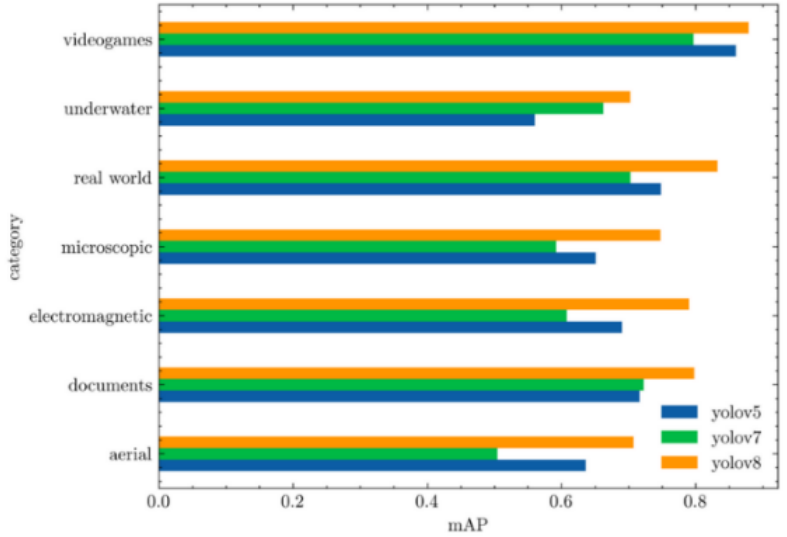


Model

- YOLOv8
  - Performance[1]



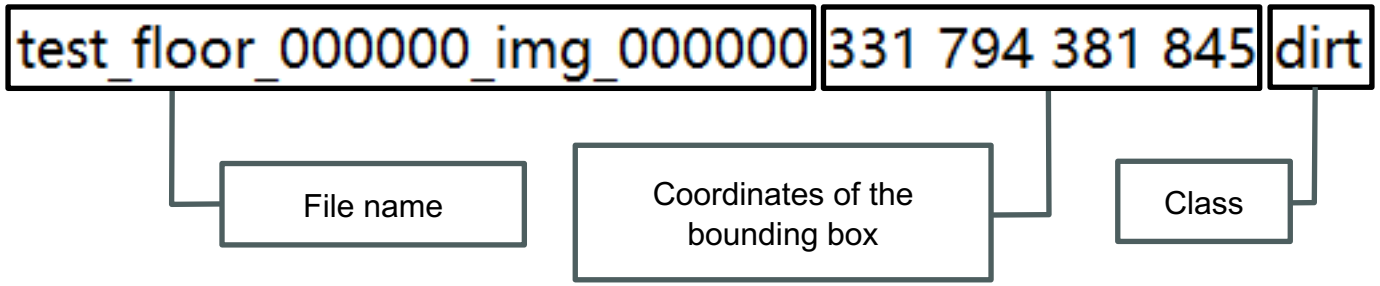
(a) YOLOs mAP@.50 against RF100 [16]



(b) YOLOs average mAP@.50 against RF100 categories

Dataset transformation

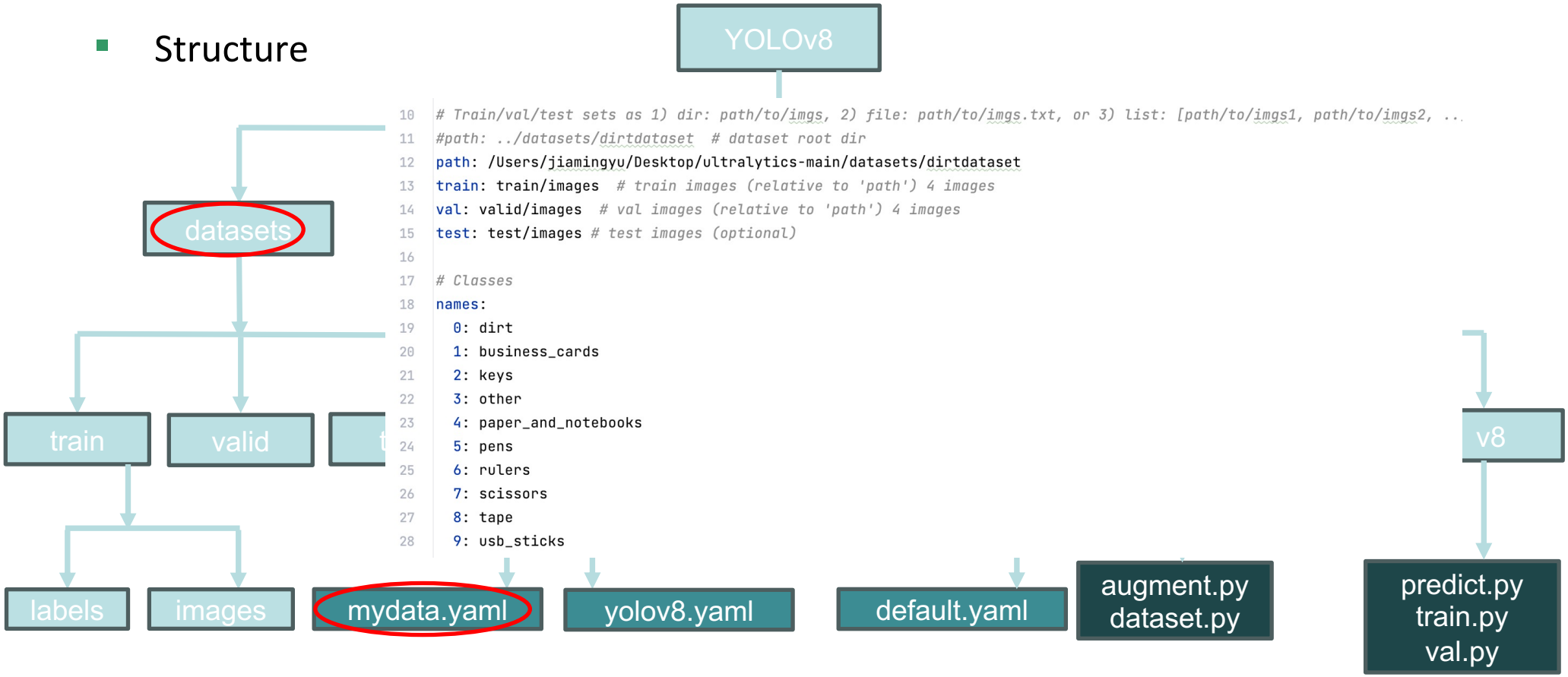
- Original label format:



- Label format in YOLO:

YOLOv8 Implementation

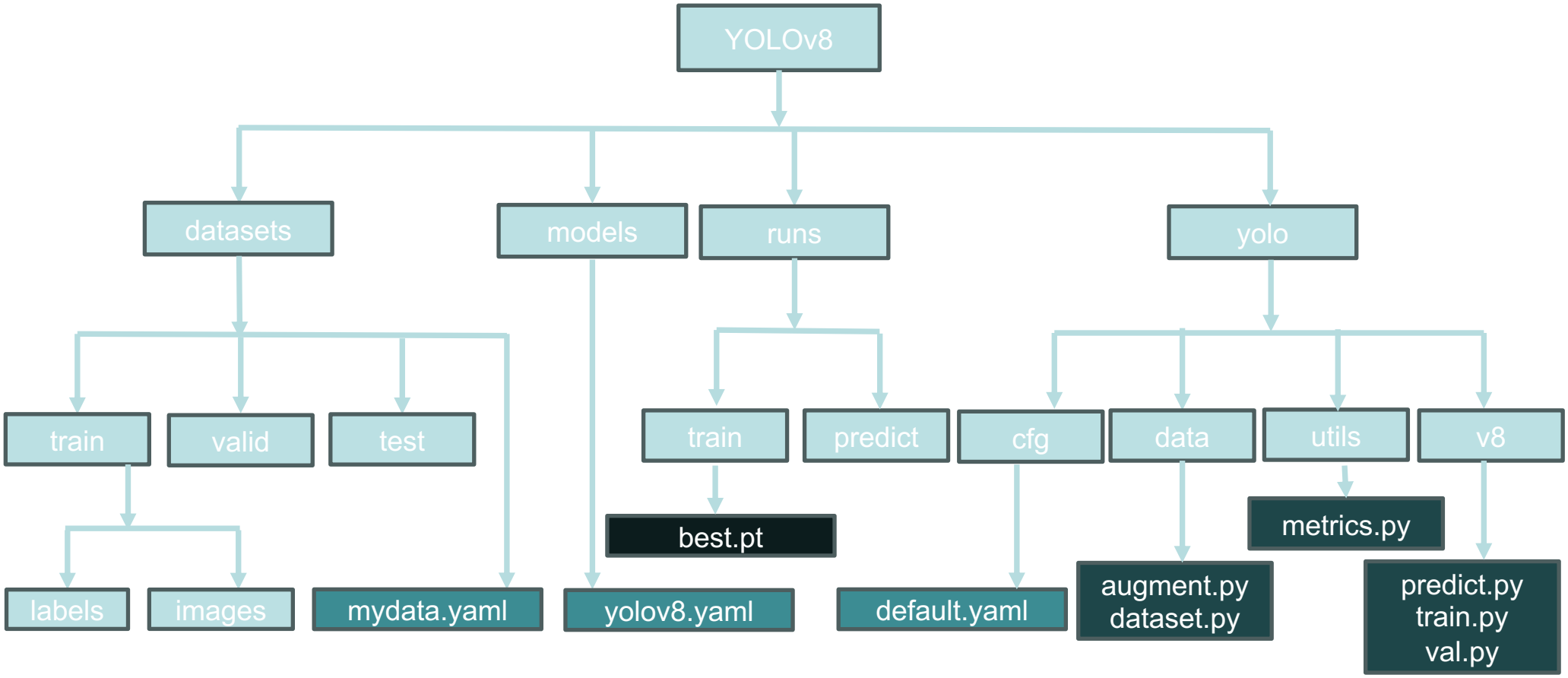
- Structure



- Train

```
yolo train data=/datasets/mydata.yaml model=yolov8n.yaml pretrained=yolov8n.pt epochs=3 batch=8 lr0=0.01
```

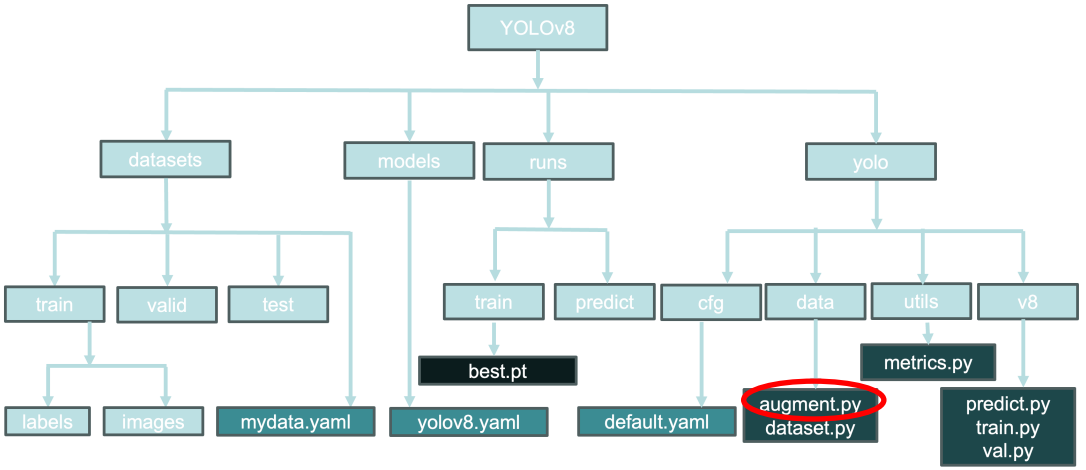
YOLOv8 Implementation



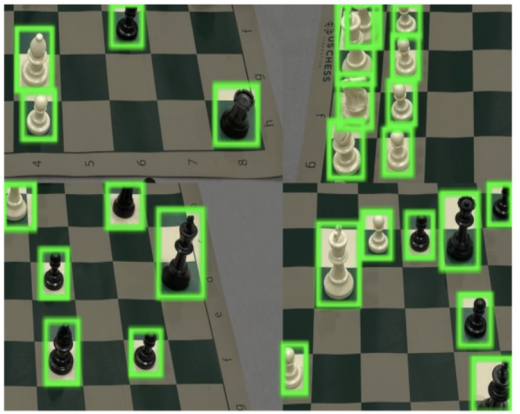
- Predict

```
yolo predict model=/runs/detect/train2/weights/best.pt source=/assets/floor1.png
```

Data Augmentation-Mosaic



Mosaic: /ultralytics/yolo/data/augment.py

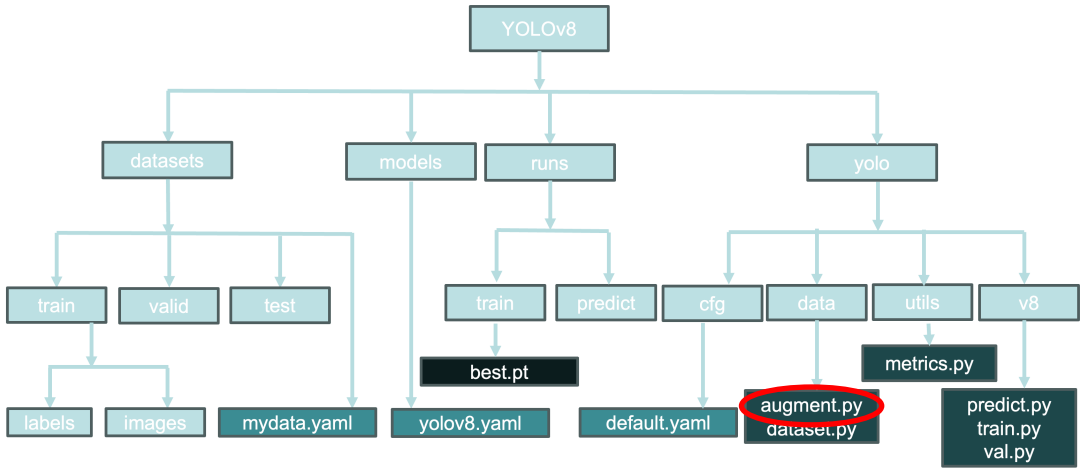


Mosaic augmentation of chess board photos

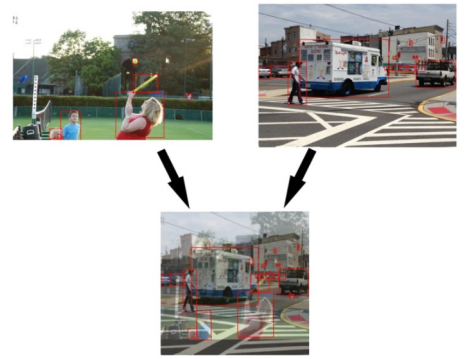
```python
class Mosaic(BaseMixTransform):
    """
    Mosaic augmentation.

    This class performs mosaic augmentation by combining multiple (4 or 9) images into a single mosaic image.
    The augmentation is applied to a dataset with a given probability.

    Attributes:
        dataset: The dataset on which the mosaic augmentation is applied.
        imgsz (int, optional): Image size (height and width) after mosaic pipeline of a single image. Default to 640.
        p (float, optional): Probability of applying the mosaic augmentation. Must be in the range 0-1. Default to 1.0
        n (int, optional): The grid size, either 4 (for 2x2) or 9 (for 3x3).
    """

    def __init__(self, dataset, imgsz=640, p=1.0, n=4):
        """Initializes the object with a dataset, image size, probability, and border."""
        assert 0 <= p <= 1.0, f'The probability should be in range [0, 1], but got {p}.'
        assert n in (4, 9), 'grid must be equal to 4 or 9.'
        super().__init__(dataset=dataset, p=p)
        self.dataset = dataset
        self.imgsz = imgsz
        self.border = (-imgsz // 2, -imgsz // 2)  # width, height
        self.n = n

    1 usage (1 dynamic)
    def get_indexes(self, buffer=True):
        """Return a list of random indexes from the dataset."""
        if buffer:  # select images from buffer
            return random.choices(list(self.dataset.buffer), k=self.n - 1)
        else:  # select any images
            return [random.randint(0, len(self.dataset) - 1) for _ in range(self.n - 1)]

    def _mix_transform(self, labels):
        """Apply mixup transformation to the input image and labels."""
        assert labels.get('rect_shape', None) is None, 'rect and mosaic are mutually exclusive.'
        assert len(labels.get('mix_labels', [])), 'There are no other images for mosaic augment.'
        return self._mosaic4(labels) if self.n == 4 else self._mosaic9(labels)
```

```python
def _mosaic4(self, labels):
    """Create a 2x2 image mosaic."""
    mosaic_labels = []
    s = self.imgsz
    yc, xc = (int(random.uniform(-x, 2 * s + x)) for x in self.border)  # mosaic center x, y
    for i in range(4):
        labels_patch = labels if i == 0 else labels['mix_labels'][i - 1]
        # Load image
        img = labels_patch['img']
        h, w = labels_patch.pop('resized_shape')

        # Place img in img4
        if i == 0:  # top left
            img4 = np.full((s * 2, s * 2, img.shape[2]), 114, dtype=np.uint8)  # base image with 4 tiles
            x1a, y1a, x2a, y2a = max(xc - w, 0), max(yc - h, 0), xc, yc  # xmin, ymin, xmax, ymax (large image)
            x1b, y1b, x2b, y2b = w - (x2a - x1a), h - (y2a - y1a), w, h  # xmin, ymin, xmax, ymax (small image)
        elif i == 1:  # top right
            x1a, y1a, x2a, y2a = xc, max(yc - h, 0), min(xc + w, s * 2), yc
            x1b, y1b, x2b, y2b = 0, h - (y2a - y1a), min(w, x2a - x1a), h
        elif i == 2:  # bottom left
            x1a, y1a, x2a, y2a = max(xc - w, 0), yc, xc, min(s * 2, yc + h)
            x1b, y1b, x2b, y2b = w - (x2a - x1a), 0, w, min(y2a - y1a, h)
        elif i == 3:  # bottom right
            x1a, y1a, x2a, y2a = xc, yc, min(xc + w, s * 2), min(s * 2, yc + h)
            x1b, y1b, x2b, y2b = 0, 0, min(w, x2a - x1a), min(y2a - y1a, h)

        img4[y1a:y2a, x1a:x2a] = img[y1b:y2b, x1b:x2b]  # img4[ymin:ymax, xmin:xmax]
        padw = x1a - x1b
        padh = y1a - y1b

        labels_patch = self._update_labels(labels_patch, padw, padh)
        mosaic_labels.append(labels_patch)
    final_labels = self._cat_labels(mosaic_labels)
    final_labels['img'] = img4
    return final_labels
```

Data Augmentation-Mixup

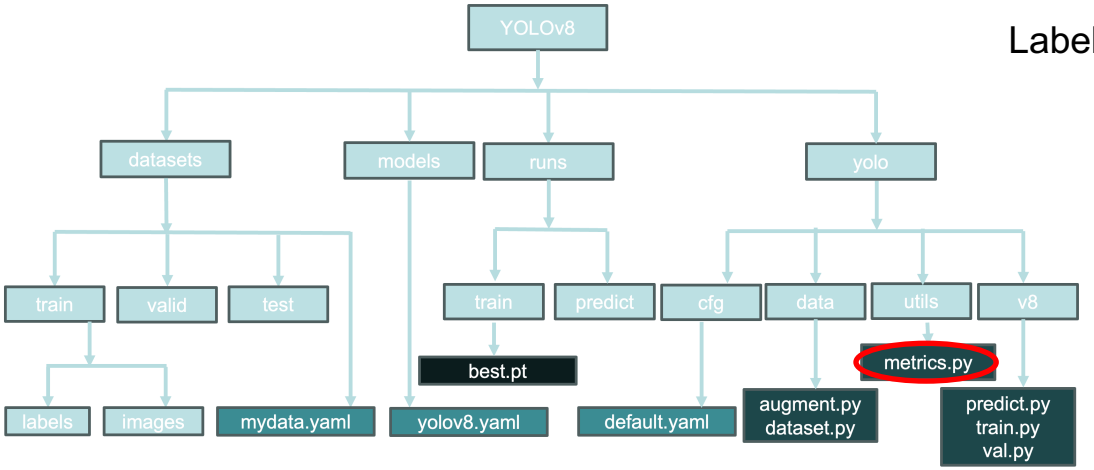


Mixup: /ultralytics/yolo/data/augment.py




```python
class MixUp(BaseMixTransform):

    def __init__(self, dataset, pre_transform=None, p=0.0) -> None:
        super().__init__(dataset=dataset, pre_transform=pre_transform, p=p)

    1 usage (1 dynamic)
    def get_indexes(self):
        """Get a random index from the dataset."""
        return random.randint(0, len(self.dataset) - 1)

    def _mix_transform(self, labels):
        """Applies MixUp augmentation https://arxiv.org/pdf/1710.09412.pdf."""
        r = np.random.beta(32.0, 32.0)  # mixup ratio, alpha=beta=32.0
        labels2 = labels['mix_labels'][0]
        labels['img'] = (labels['img'] * r + labels2['img'] * (1 - r)).astype(np.uint8)
        labels['instances'] = Instances.concatenate([labels['instances'], labels2['instances']], axis=0)
        labels['cls'] = np.concatenate([labels['cls'], labels2['cls']], 0)
        return labels
```

Data Augmentation-Label Smoothing



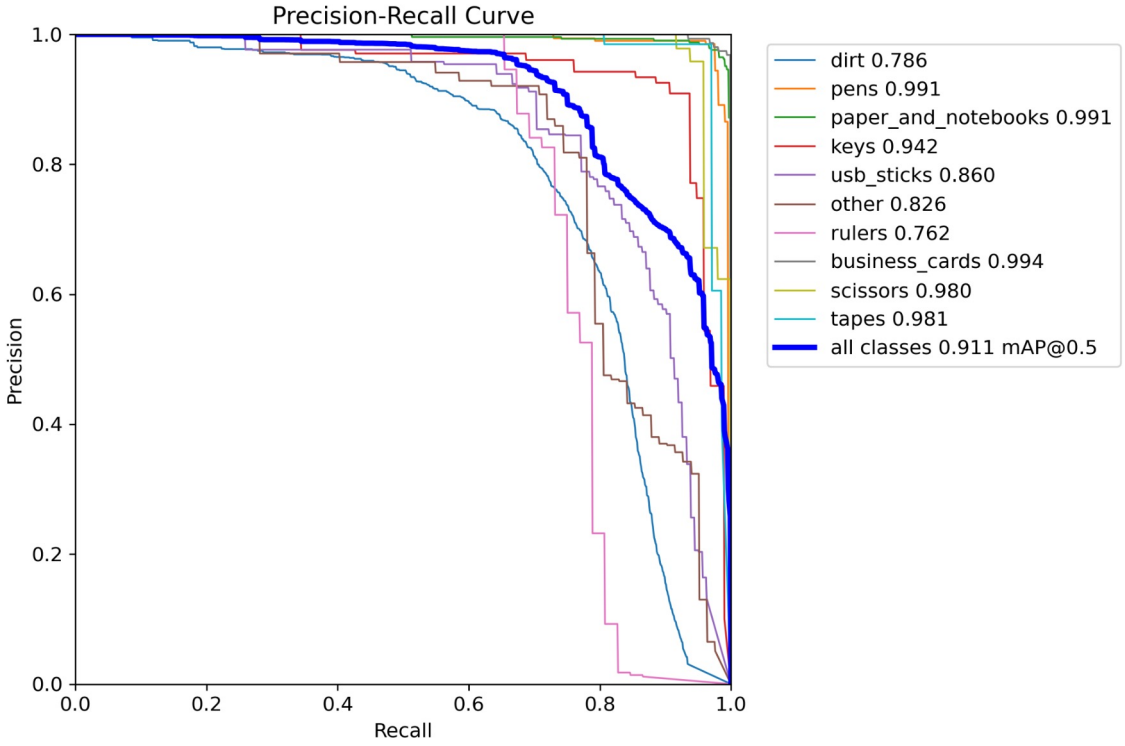
Label Smoothing: /ultralytics/yolo/utils/metrics.py

```
173  def smooth_BCE(eps=0.1):  # https://github.com/ultralytics/yolov3/issues/238#issuecomment-598028441
174      # return positive, negative label smoothing BCE targets
175      return 1.0 - 0.5 * eps, 0.5 * eps
```

$$\text{label}_{\text{true}}^{\text{smooth}} = \text{label}_{\text{true}} \times (1 - \alpha) + \text{label}_{\text{true}} \times \alpha$$

Results of Original Version

- Training with 1156 pictures
  - yolo train data=C:\Users\c1257\Desktop\ultralytics-main\datasets\yolov8dirt.yaml model=yolov8s.yaml pretrained=yolov8s.pt epochs=30 batch=8 lr0=0.0 mosaic=0



Hyperparameters Tuning

Lots of choices to find a better detection results

```
# 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: 0.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

Tuning Mosaic

- Epoch=30 Batch=8 (1156 Training Set)

Mosaic	all classes mAP	dirt	pens	paper	keys	usb	rulers	card	scissors	tapes	others
0	0.911	0.786	0.991	0.991	0.942	0.86	0.762	0.994	0.98	0.981	0.826
0.3	0.902	0.767	0.991	0.991	0.927	0.851	0.727	0.994	0.987	0.99	0.798
0.5	0.915	0.785	0.99	0.995	0.949	0.877	0.724	0.993	0.993	0.982	0.858
0.7	0.907	0.78	0.988	0.994	0.938	0.856	0.755	0.976	0.974	0.988	0.823
1	0.914	0.804	0.989	0.991	0.948	0.878	0.732	0.994	0.975	0.98	0.85

Tuning Mixup

- Epoch=30 Batch=8 Mosaic=1 (1156 Training Set)

Mixup	all classes mAP	dirt	pens	paper	keys	usb	rulers	card	scissors	tapes	others
0	0.914	0.804	0.989	0.991	0.948	0.878	0.732	0.994	0.975	0.98	0.85
0.1	0.914	0.78	0.988	0.992	0.958	0.86	0.771	0.994	0.963	0.984	0.847
0.2	0.906	0.779	0.984	0.992	0.923	0.857	0.742	0.994	0.976	0.987	0.826
0.3	0.914	0.8	0.988	0.993	0.946	0.874	0.754	0.995	0.991	0.973	0.828
0.4	0.911	0.795	0.988	0.993	0.958	0.867	0.74	0.994	0.99	0.971	0.813
0.5	0.914	0.782	0.983	0.994	0.941	0.867	0.746	0.994	0.981	0.989	0.859
0.7	0.907	0.783	0.988	0.995	0.951	0.866	0.716	0.989	0.993	0.971	0.817
0.9	0.912	0.808	0.983	0.991	0.939	0.868	0.716	0.994	0.995	0.986	0.839

Tuning Label Smoothing

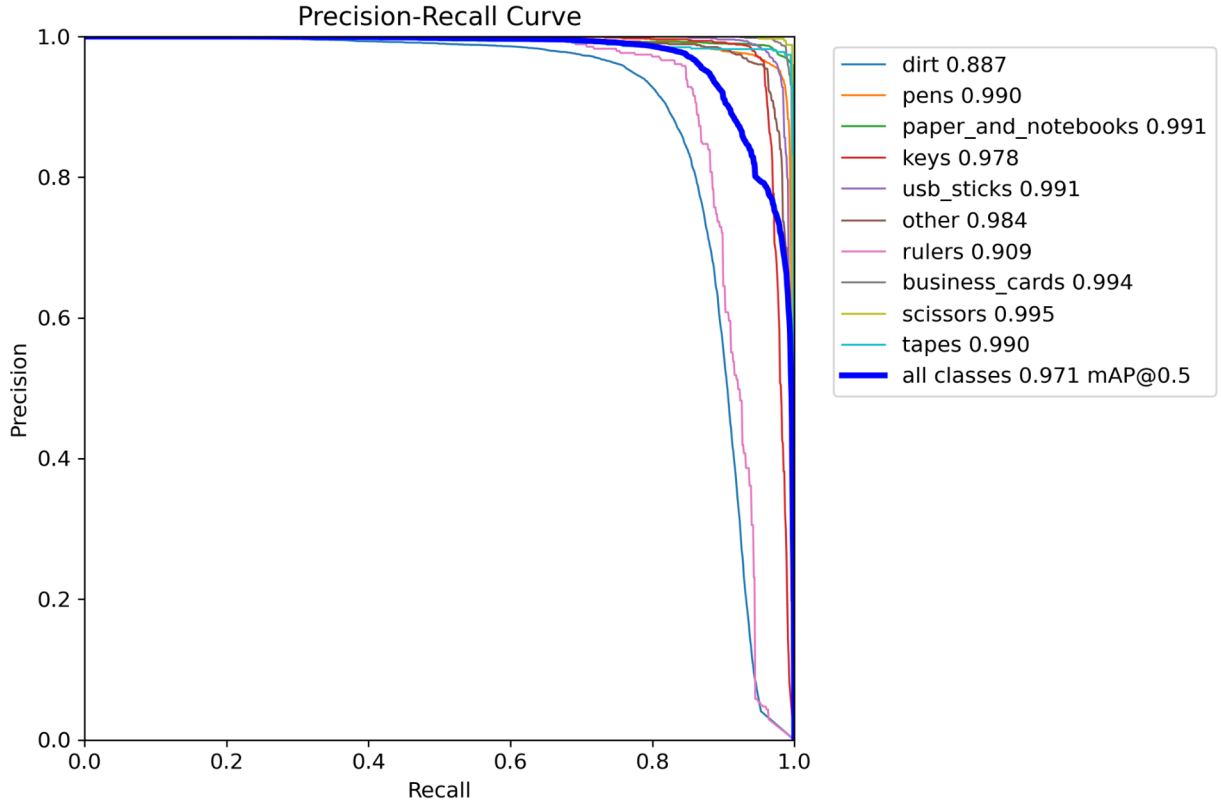
- Epoch=30 Batch=8 Mosaic=1 (1156 Training Set)

Label-Smoothing	all classes mAP	dirt	pens	paper	keys	usb	rulers	card	scissors	tapes	others
0	0.914	0.804	0.989	0.991	0.948	0.878	0.732	0.994	0.975	0.98	0.85
0.1	0.914	0.804	0.989	0.991	0.948	0.878	0.732	0.994	0.975	0.98	0.85
0.3	0.914	0.804	0.989	0.991	0.948	0.878	0.732	0.994	0.975	0.98	0.85

Mixup=0.3 + Mosaic=1 is so far the best version!

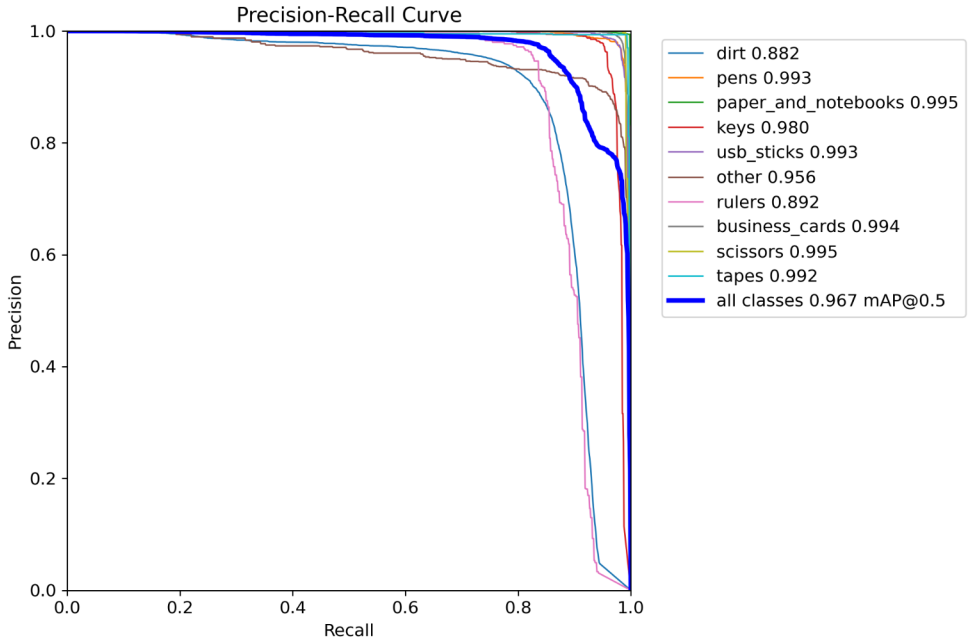
Final Training

- Training with larger dataset with more epoches
- Epoch=100 Batch=16 Mosaic=1 Mixup=0.3 (9248 Training Set 3448 Test)

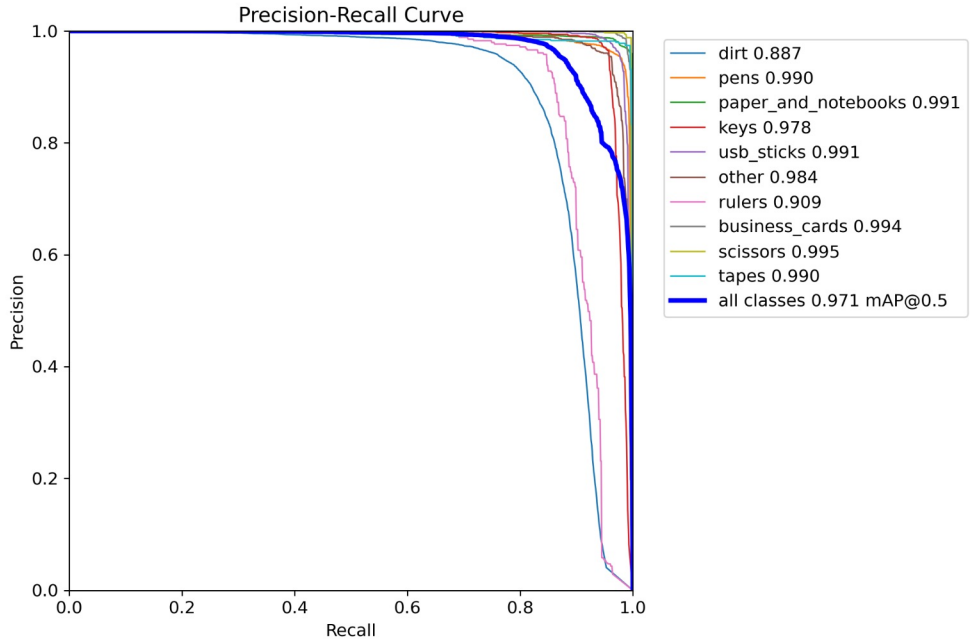


Comparison before and after

Comparison based on 9248 data set and 100 epoches



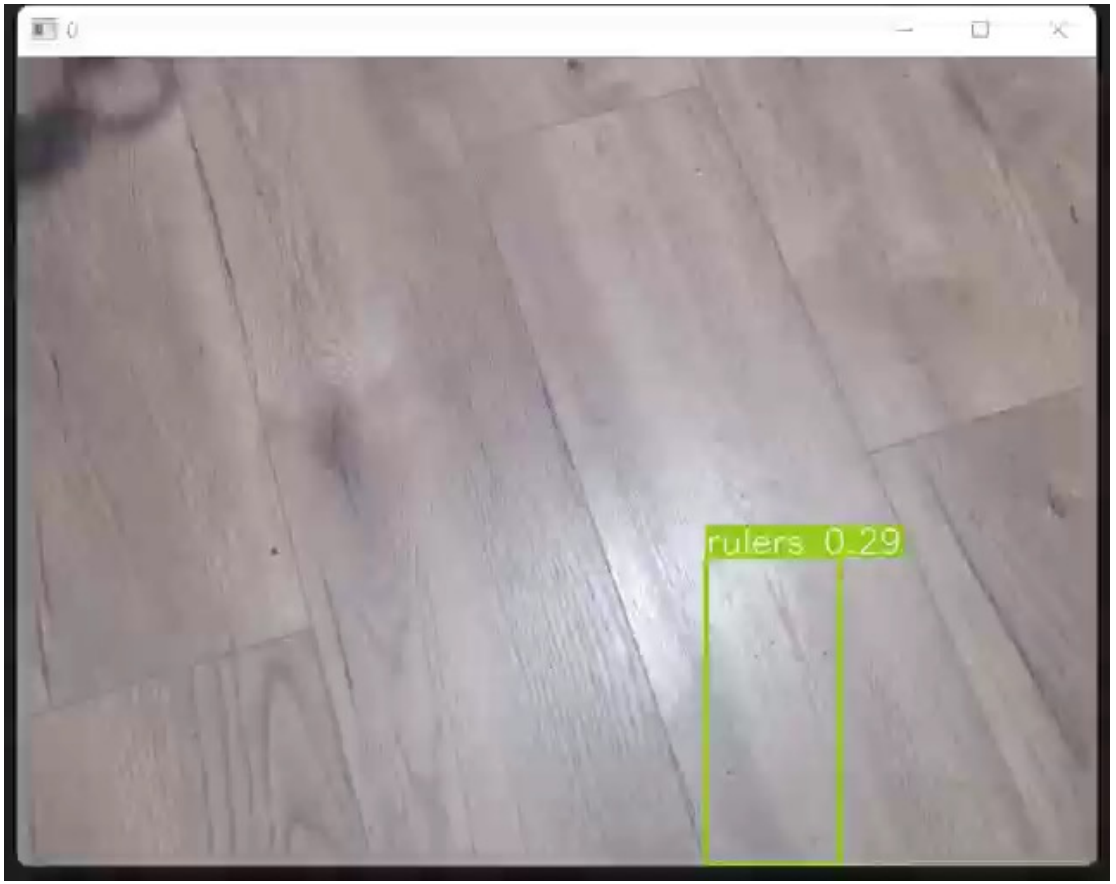
before tuning



after tuning

Further Applications with optimized data

- Using the file “best.pt” from the final training and apply it to the real_cam

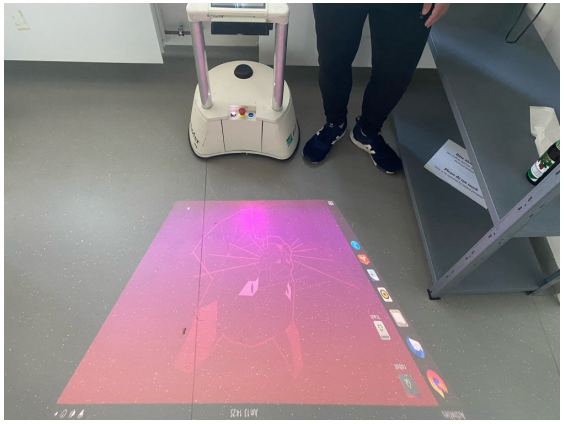


Summary and Outlook

- Summary of the project
  - Reached mAP for all classes of 0.971
  - Processing speed about 67 FPS

- Major factors
  - Imbalanced samples of different class
    - Solution: Data augmentation
  - Regularization
    - Label smoothing

- Outlook
  - Transplant the model to ROS



Reference

- [1] Reis D, Kupec J, Hong J, et al. Real-Time Flying Object Detection with YOLOv8[J]. arXiv preprint arXiv:2305.09972, 2023.

- [2] Ultralytics. GitHub - ultralytics/ultralytics: NEW - YOLOv8 in PyTorch > ONNX > CoreML > TFLite. GitHub. https://github.com/ultralytics/ultralytics.

End

Thank you for your attention!