



Random Interpolation Resize: A free image data augmentation method for object detection in industry

School of Instrument Science and Opto-electronic Engineering, Hefei University of Technology, Hefei 230009, China

Dahang Wan, Rongsheng Lu *, Ting Xu, Siyuan Shen, Xianli Lang, Zhijie Ren



论文链接: <https://www.sciencedirect.com/science/article/pii/S0957417423008576>

开源地址: <https://github.com/wandahangFY/RIR> .

1.论文动机：从插值方式的角度进行数据增强（详情请看原文）

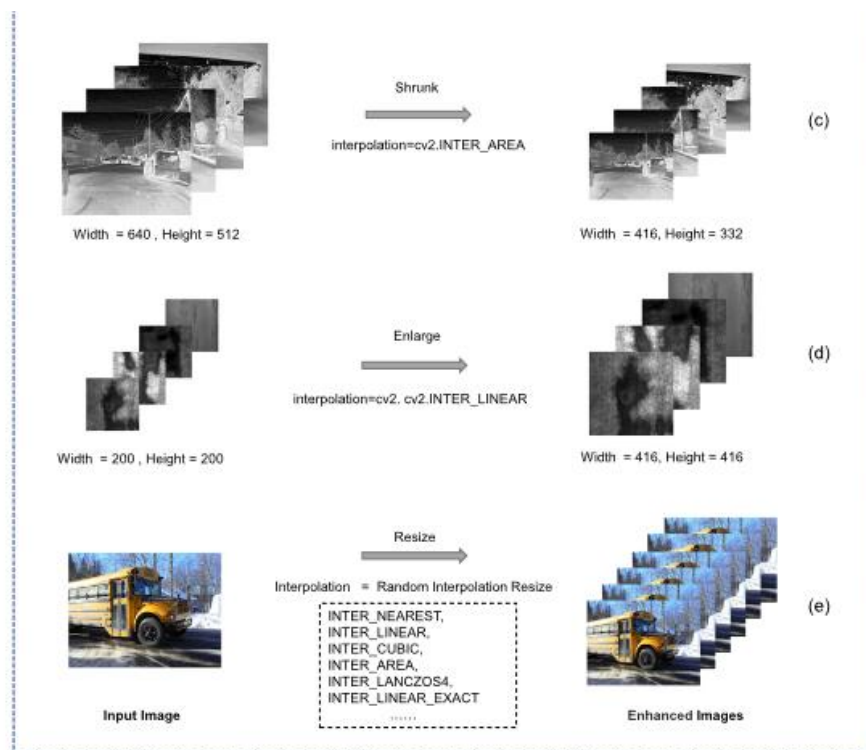


Fig. 2. Comparison of different methods of resize.

不使用RIR 方法：多个epoch迭代，每张图片只有一种插值方式

使用RIR 方法：多次迭代，每张图片可以有不同的插值方式

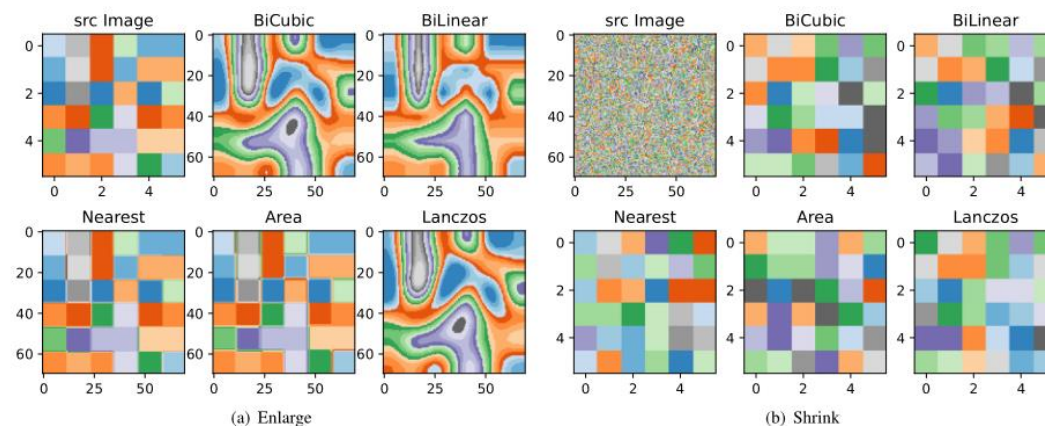


Fig. 3. Comparison of the results of various interpolation methods.

图3采用不同的插值方式对原图（src Image）进行放大或缩小从图中可以看出，不同的插值方式之间是有差异的

2.原理：在训练阶段随机使用插值方式，在测试阶段采用默认的插值方式

训练阶段

使用RIR 方法

```
interp = random_interpolation_resize(  
    cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)
```

使用RIR 方法

验证阶段

不使用RIR 方法

(采用常规的插值方式)

(YOLOv8, 11月之前有两种插值方式,
11月以后全部变成了双线性插值)

```
interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
```

```
# -----rir start-----  
if self.use_rir:  
    if self.val_flag:  
        interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA  
        # print("use_rir={},val_flag={}".format(self.use_rir,self.val_flag))  
    else:  
        interp = random_interpolation_resize(  
            cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)  
        # print("use_rir={},val_flag={}".format(self.use_rir, self.val_flag))  
    else:  
        interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA  
im = cv2.resize(im,  
    (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz)),  
    interpolation=interp)  
# -----rir end-----
```

3.添加教程

3.1 YOLOv8 添加步骤(已完成)

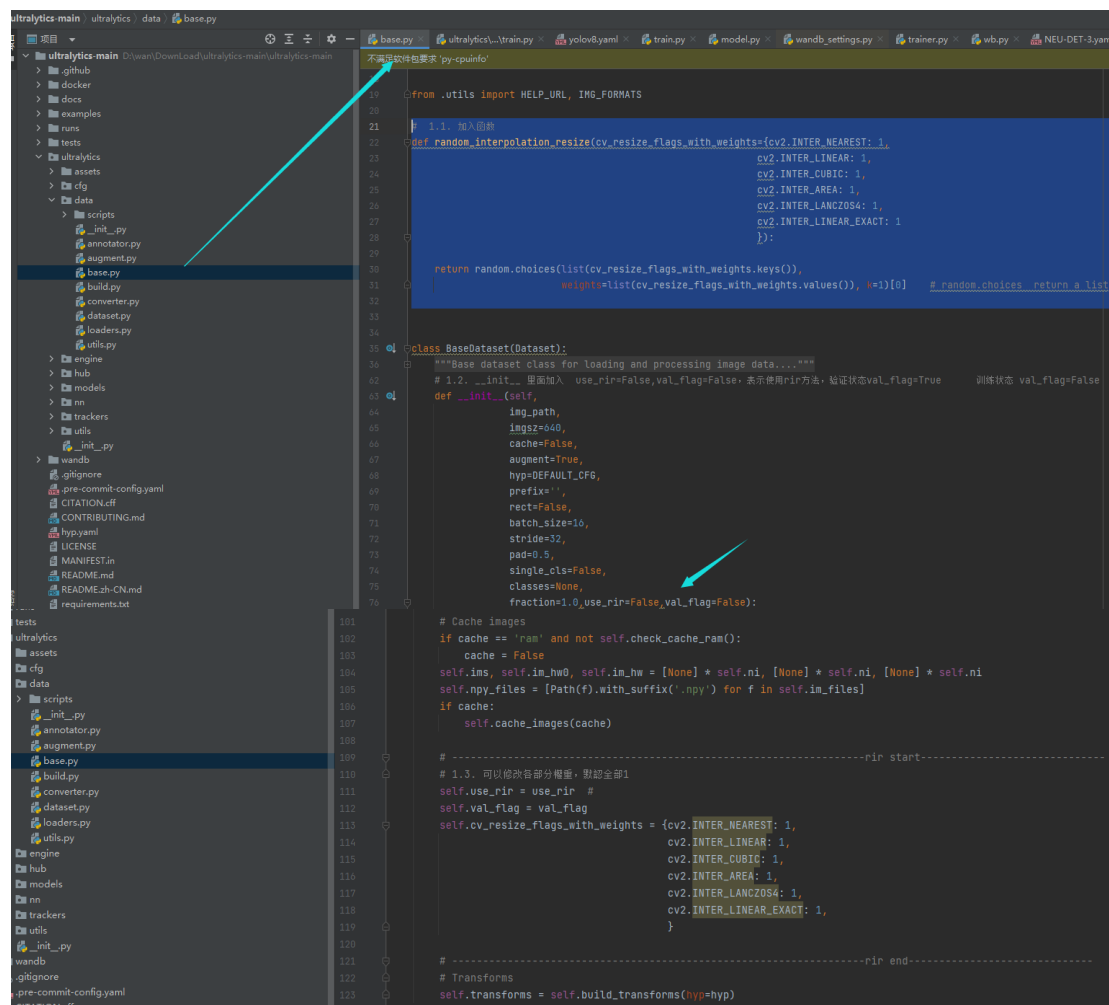
3.2 YOLOv5 添加步骤(TODO)

3.3 YOLOv7 添加步骤(TODO)

3.4 YOLOv5超参数进化 添加步骤(TODO)

3.1 YOLOv8 添加步骤

1. base.py 内部更改



ultralytics/data/base.py 11月以后版本 (最新版)

ultralytics/yolo/data/base.py 11月以前的版本

1.1. 加入函数

```
def random_interpolation_resize(cv_resize_flags_with_weights={cv2.INTER_NEAREST: 1,
cv2.INTER_LINEAR: 1,
cv2.INTER_CUBIC: 1,
cv2.INTER_AREA: 1,
cv2.INTER_LANCZOS4: 1,
cv2.INTER_LINEAR_EXACT: 1
}):
    return random.choices(list(cv_resize_flags_with_weights.keys()),
weights=list(cv_resize_flags_with_weights.values()), k=1)[0] # random.choices return a list
```

1.2. __init__ 里面加入 use_rir=False, val_flag=False 表示使用rir方法, 验证状态val_flag=True 训练状态 val_flag=False

```
# -----rir start-----
# 1.3. 引入相关参数, 可以修改各部分权重, 默认全部1
self.use_rir = use_rir #
self.val_flag = val_flag
self.cv_resize_flags_with_weights = {cv2.INTER_NEAREST: 1,
cv2.INTER_LINEAR: 1,
cv2.INTER_CUBIC: 1,
cv2.INTER_AREA: 1,
cv2.INTER_LANCZOS4: 1,
cv2.INTER_LINEAR_EXACT: 1,
}
# -----rir end-----
```

3.1 YOLOv8 添加步骤

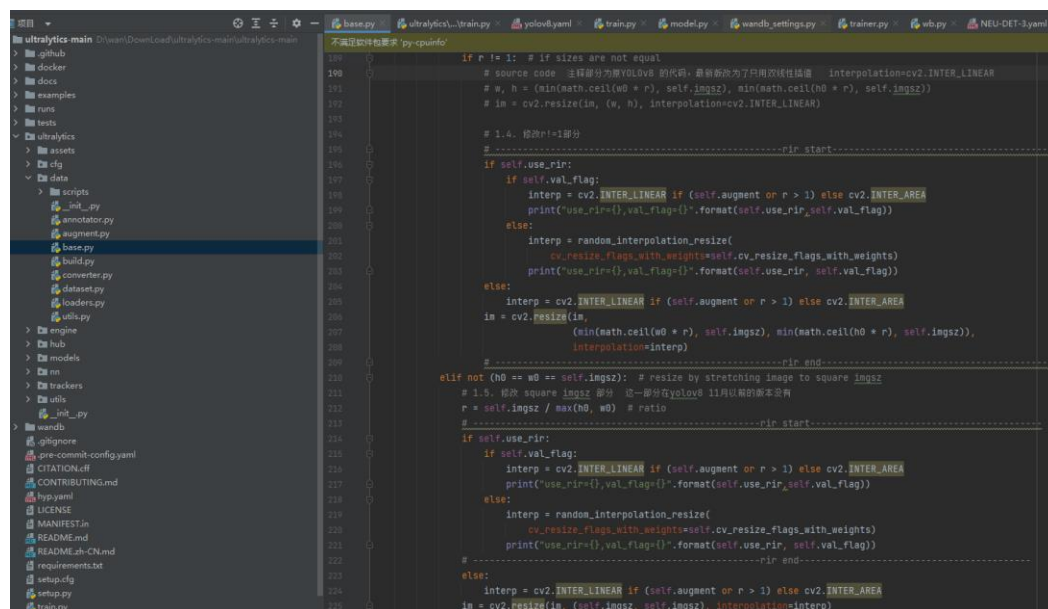
1. base.py 内部更改(11月以后)

1.4. 修改r!=1部分

1.5. 修改 square imgsiz 部分 这一部分在YOLOv8 11月以前的版本没有

```
class BaseDataset(Dataset):
```

```
def load_image(self, i, rect_mode=True):
```



```
190 if r != 1: # if sizes are not equal
191     # source code 注释部分为原YOLOv8 的代码，最新版改为了只用双线性插值 interpolation=cv2.INTER_LINEAR
192     # w, h = (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz))
193     # im = cv2.resize(im, (w, h), interpolation=cv2.INTER_LINEAR)
194
195     # 1.4. 修改r!=1部分
196     # -----rir start-----
197     if self.use_rir:
198         if self.val_flag:
199             interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
200             print("use_rir={},val_flag={}".format(self.use_rir,self.val_flag))
201         else:
202             interp = random_interpolation_resize(
203                 cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)
204             print("use_rir={},val_flag={}".format(self.use_rir, self.val_flag))
205     else:
206         interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
207         in = cv2.resize(im, (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz)),
208             interpolation=interp)
209
210     # -----rir end-----
211 elif not (h0 == w0 == self.imgsz): # resize by stretching image to square imgsiz
212     # 1.5. 修改 square imgsiz 部分 这一部分在yolov8 11月以前的版本没有
213     r = self.imgsz / max(h0, w0) # ratio
214     # -----rir start-----
215     if self.use_rir:
216         if self.val_flag:
217             interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
218             print("use_rir={},val_flag={}".format(self.use_rir,self.val_flag))
219         else:
220             interp = random_interpolation_resize(
221                 cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)
222             print("use_rir={},val_flag={}".format(self.use_rir, self.val_flag))
223     else:
224         interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
225         in = cv2.resize(im, (self.imgsz, self.imgsz), interpolation=interp)
```

```
if rect_mode: # resize long side to imgsiz while maintaining aspect ratio
    r = self.imgsz / max(h0, w0) # ratio
    if r != 1: # if sizes are not equal
        # source code 注释部分为原YOLOv8 的代码，最新版改为了只用双线性插值
        interpolation=cv2.INTER_LINEAR
        # w, h = (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz))
        # im = cv2.resize(im, (w, h), interpolation=cv2.INTER_LINEAR)
```

1.4. 修改r!=1部分

```
# -----rir start-----
```

```
if self.use_rir:
    if self.val_flag:
        interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
        print("use_rir={},val_flag={}".format(self.use_rir,self.val_flag))
    else:
        interp = random_interpolation_resize(
            cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)
        print("use_rir={},val_flag={}".format(self.use_rir, self.val_flag))
else:
    interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
    im = cv2.resize(im, (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz)),
        interpolation=interp)
```

```
# -----rir end-----
```

```
elif not (h0 == w0 == self.imgsz): # resize by stretching image to square imgsiz
```

```
# 1.5. 修改 square imgsiz 部分 这一部分在yolov8 11月以前的版本没有
```

```
r = self.imgsz / max(h0, w0) # ratio
```

```
# -----rir start-----
```

```
if self.use_rir:
    if self.val_flag:
        interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
        print("use_rir={},val_flag={}".format(self.use_rir,self.val_flag))
    else:
        interp = random_interpolation_resize(
            cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)
        print("use_rir={},val_flag={}".format(self.use_rir, self.val_flag))
else:
    interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
    im = cv2.resize(im, (self.imgsz, self.imgsz), interpolation=interp)
```

```
# -----rir end-----
```

```
else:
    interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
    im = cv2.resize(im, (self.imgsz, self.imgsz), interpolation=interp)
```

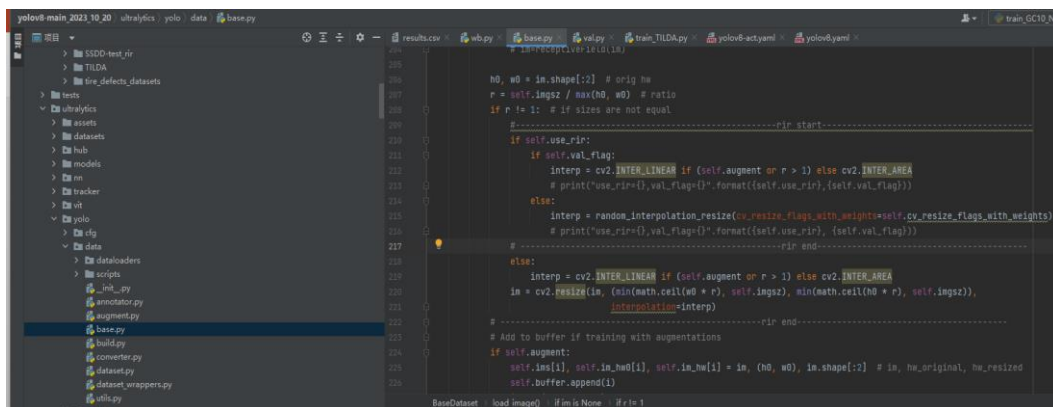
3.1 YOLOv8 添加步骤

1. base.py 内部更改(11月以前的版本)

1.4. 修改r!=1部分

```
class BaseDataset(Dataset):
```

```
def load_image(self, i, rect_mode=True):
```



```
if r != 1: # if sizes are not equal
#-----rir start-----
if self.use_rir:
    if self.val_flag:
        interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
        # print("use_rir={},val_flag={}".format({self.use_rir},{self.val_flag}))
    else:
        interp =
random_interpolation_resize(cv_resize_flags_with_weights=self.cv_resize_flags_with_weights
)
        # print("use_rir={},val_flag={}".format({self.use_rir},{self.val_flag}))
# -----rir end-----
else:
    interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
    im = cv2.resize(im, (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz)),
                    interpolation=interp)
# -----rir end-----
```


3.1 YOLOv8 添加步骤

1. base.py 内部更改(11月以后的版本)

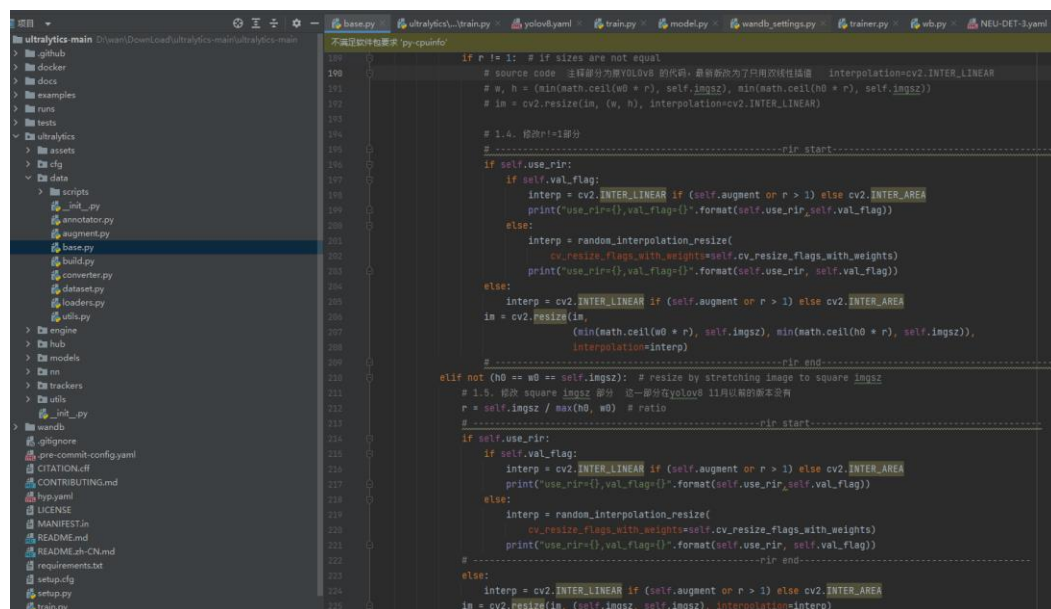
```
if rect_mode:
```

```
class BaseDataset(Dataset):
```

```
def load_image(self, i, rect_mode=True):
```

1.4. 修改r!=1部分

1.5. 修改 square imgsiz 部分 这一部分在YOLOv8 11月以前的版本没有



```
if rect_mode: # resize long side to imgsiz while maintaining aspect ratio
    r = self.imgsz / max(h0, w0) # ratio
    if r != 1: # if sizes are not equal
        # source code 注释部分为原YOLOv8 的代码，最新版改为了只用双线性插值
        interpolation=cv2.INTER_LINEAR
        # w, h = (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz))
        # im = cv2.resize(im, (w, h), interpolation=cv2.INTER_LINEAR)
```

1.4. 修改r!=1部分

-----rir start-----

```
if self.use_rir:
    if self.val_flag:
        interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
        print("use_rir={},val_flag={}".format(self.use_rir,self.val_flag))
    else:
        interp = random_interpolation_resize(
            cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)
        print("use_rir={},val_flag={}".format(self.use_rir, self.val_flag))
else:
```

```
    interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
    im = cv2.resize(im,
                    (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0 * r), self.imgsz)),
                    interpolation=interp)
```

-----rir end-----

```
elif not (h0 == w0 == self.imgsz): # resize by stretching image to square imgsiz
```

1.5. 修改 square imgsiz 部分 这一部分在yolov8 11月以前的版本没有

r = self.imgsz / max(h0, w0) # ratio

-----rir start-----

```
if self.use_rir:
    if self.val_flag:
        interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
        print("use_rir={},val_flag={}".format(self.use_rir,self.val_flag))
    else:
        interp = random_interpolation_resize(
            cv_resize_flags_with_weights=self.cv_resize_flags_with_weights)
        print("use_rir={},val_flag={}".format(self.use_rir, self.val_flag))
else:
```

-----rir end-----

```
    interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
    im = cv2.resize(im, (self.imgsz, self.imgsz), interpolation=interp)
```


3.1 YOLOv8 添加步骤

2. build.py build_yolo_dataset

ultralytics/data/build.py

ultralytics/yolo/data/build.py

```
def build_yolo_dataset(cfg, img_path, batch, data, mode='train', rect=False, stride=32, use_rir=False):  
    """Build YOLO Dataset."""  
    return YOLODataset(  
        img_path=img_path,  
        imgs=cfg.imgs,  
        batch_size=batch,  
        augment=mode == 'train', # augmentation  
        hyp=cfg, # TODO: probably add a get_hyps_from_cfg function  
        rect=cfg.rect or rect, # rectangular batches  
        cache=cfg.cache or None,  
        single_cls=cfg.single_cls or False,  
        stride=int(stride),  
        pad=0.0 if mode == 'train' else 0.5,  
        prefix=cfg.prefix, #  
        use_segments=cfg.task == 'segment',  
        use_keypoints=cfg.task == 'pose',  
        classes=cfg.classes,  
        data=data,  
        fraction=cfg.fraction if mode == 'train' else 1.0,  
        use_rir=use_rir, # 2.2 修改 use_rir=use_rir  
        val_flag=False if mode == 'train' else True, # 2.3 修改 val_flag  
    )
```

```
# 2.1 修改 use_rir=False  
def build_yolo_dataset(cfg, img_path, batch, data, mode='train', rect=False, stride=32, use_rir=False):  
    """Build YOLO Dataset."""  
    return YOLODataset(  
        img_path=img_path,  
        imgs=cfg.imgs,  
        batch_size=batch,  
        augment=mode == 'train', # augmentation  
        hyp=cfg, # TODO: probably add a get_hyps_from_cfg function  
        rect=cfg.rect or rect, # rectangular batches  
        cache=cfg.cache or None,  
        single_cls=cfg.single_cls or False,  
        stride=int(stride),  
        pad=0.0 if mode == 'train' else 0.5,  
        prefix=cfg.prefix, #  
        use_segments=cfg.task == 'segment',  
        use_keypoints=cfg.task == 'pose',  
        classes=cfg.classes,  
        data=data,  
        fraction=cfg.fraction if mode == 'train' else 1.0,  
        use_rir=use_rir, # 2.2 修改 use_rir=use_rir  
        val_flag=False if mode == 'train' else True, # 2.3 修改 val_flag  
    )
```

3.1 YOLOv8 添加步骤

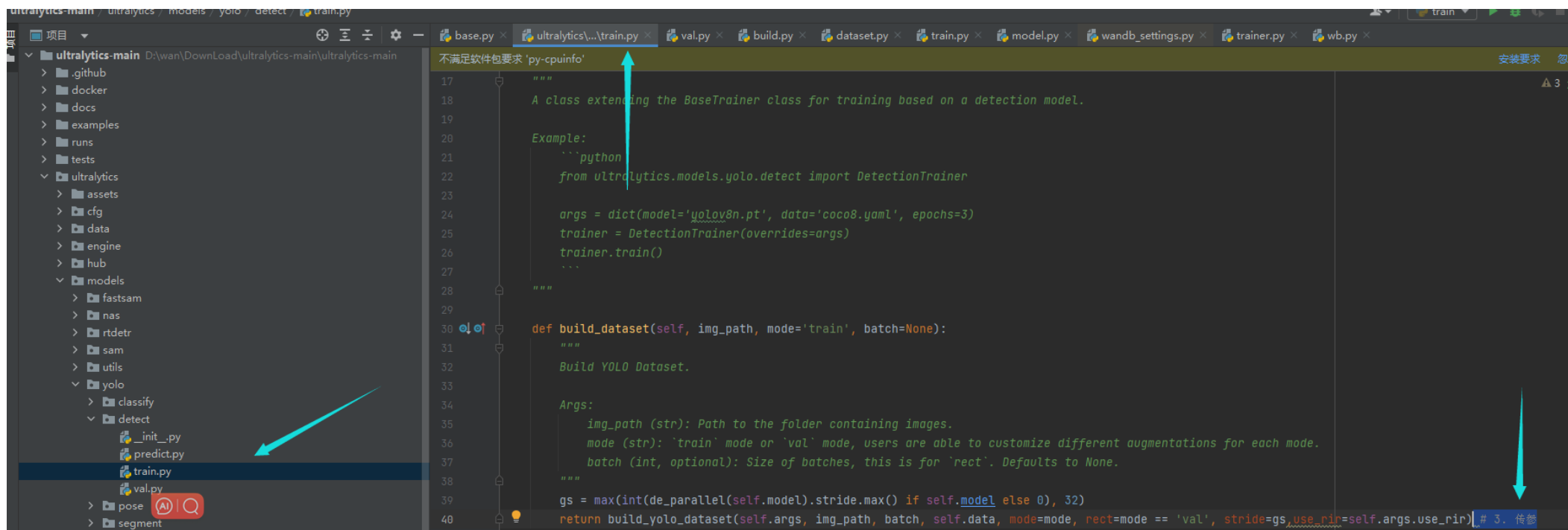
3. ultralytics/models/yolo/detect/train.py

```
class DetectionTrainer(BaseTrainer):
```

传参

```
def build_dataset(self, img_path, mode='train', batch=None):
    """
    Build YOLO Dataset.

    Args:
        img_path (str): Path to the folder containing images.
        mode (str): 'train' mode or 'val' mode, users are able to customize different augmentations for each mode.
        batch (int, optional): Size of batches, this is for 'rect'. Defaults to None.
    """
    gs = max(int(de_parallel(self.model).stride.max() if self.model else 0), 32)
    return build_yolo_dataset(self.args, img_path, batch, self.data, mode=mode, rect=mode == 'val',
                             stride=gs, use_rir=self.args.use_rir) # 3. 传参
```

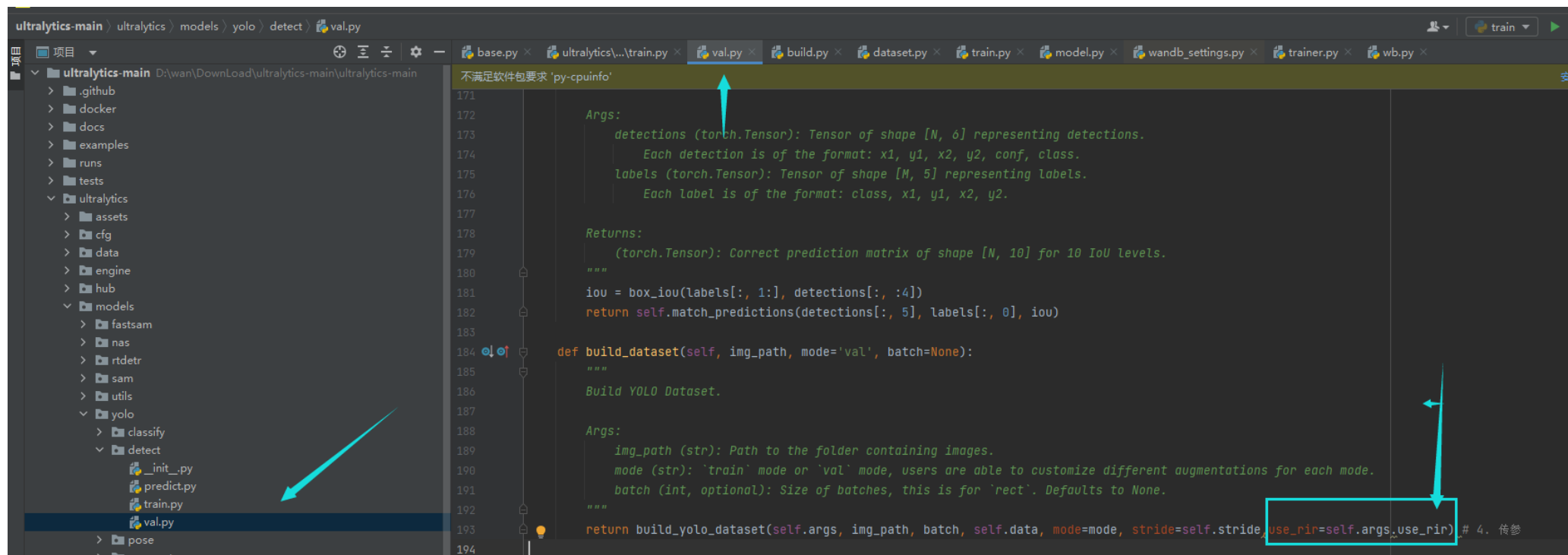


3.1 YOLOv8 添加步骤

4. ultralytics/models/yolo/detect/val.py 传参

```
class DetectionValidator(BaseValidator):
```

```
def build_dataset(self, img_path, mode='val', batch=None):  
    """  
    Build YOLO Dataset.  
  
    Args:  
        img_path (str): Path to the folder containing images.  
        mode (str): 'train' mode or 'val' mode, users are able to customize different augmentations for each mode.  
        batch (int, optional): Size of batches, this is for 'rect'. Defaults to None.  
    """  
    return build_yolo_dataset(self.args, img_path, batch, self.data, mode=mode,  
                             stride=self.stride, use_rir=self.args.use_rir) # 4. 传参
```

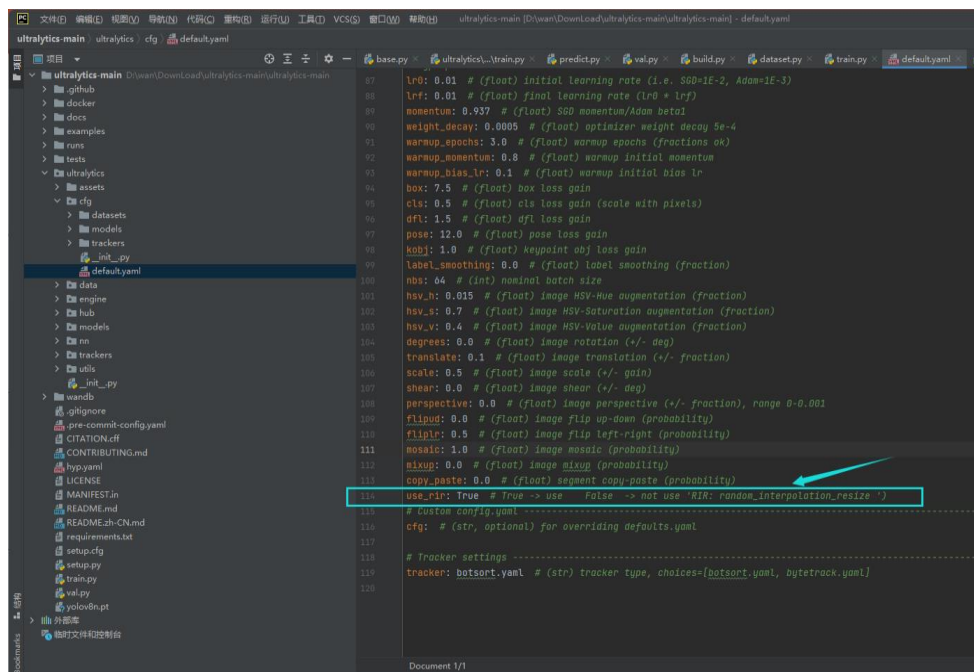


3.1 YOLOv8 添加步骤

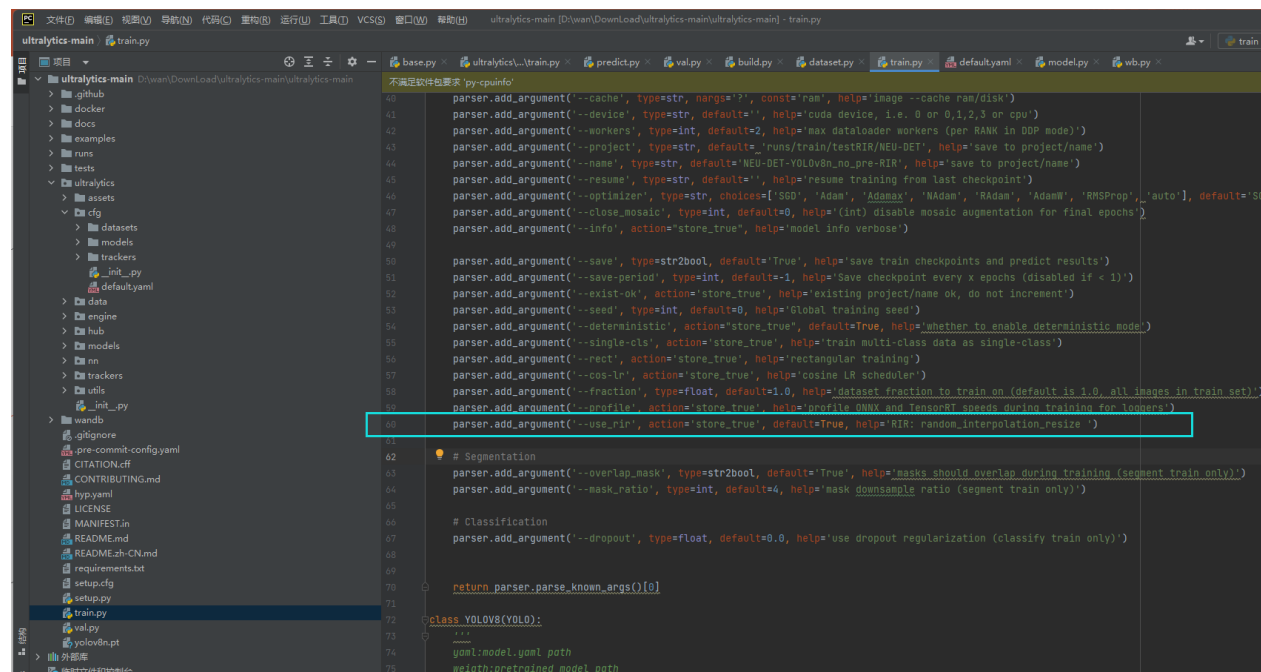
5. ultralytics/cfg/default.yaml 添加 train.py 添加（如果有的话）

```
use_rir: True # True -> use False -> not use 'RIR: random_interpolation_resize')
```

```
parser.add_argument('--use_rir', action='store_true', default=True, help='RIR: random_interpolation_resize')
```



```
lr0: 0.01 # (float) initial learning rate (i.e. SGD+1E-2, Adam+1E-3)
lr1: 0.01 # (float) final learning rate (lr0 * lr1)
momentum: 0.937 # (float) SGD momentum/Adam beta1
weight_decay: 0.0005 # (float) optimizer weight decay 5e-4
warmup_epochs: 3.0 # (float) warmup epochs (fractions ok)
warmup_momentum: 0.8 # (float) warmup initial momentum
warmup_bias_lr: 0.1 # (float) warmup initial bias lr
box: 7.5 # (float) box loss gain
cls: 0.5 # (float) cls loss gain (scale with pixels)
dfl: 1.5 # (float) dfl loss gain
pose: 12.0 # (float) pose loss gain
kobj: 1.0 # (float) keypoint obj loss gain
label_smoothing: 0.0 # (float) label smoothing (fraction)
nbs: 64 # (int) nominal batch size
hsv_h: 0.015 # (float) image HSV-Hue augmentation (fraction)
hsv_s: 0.7 # (float) image HSV-Saturation augmentation (fraction)
hsv_v: 0.4 # (float) image HSV-Value augmentation (fraction)
degrees: 0.0 # (float) image rotation (+/- deg)
translate: 0.1 # (float) image translation (+/- fraction)
scale: 0.5 # (float) image scale (+/- gain)
shear: 0.0 # (float) image shear (+/- deg)
perspective: 0.0 # (float) image perspective (+/- fraction), range 0-0.001
flipud: 0.0 # (float) image flip up-down (probability)
fliplr: 0.5 # (float) image flip left-right (probability)
mosaic: 1.0 # (float) image mosaic (probability)
mixup: 0.0 # (float) image mixup (probability)
copy_paste: 0.0 # (float) segment copy-paste (probability)
use_rir: True # True -> use False -> not use 'RIR: random_interpolation_resize'
# custom config.yaml -----
cfg: # (str, optional) for overriding defaults.yaml
# Tracker settings -----
tracker: botsort.yaml # (str) tracker type, choices=[botsort.yaml, bytetrack.yaml]
```



```
parser.add_argument('--cache', type=str, nargs='?', const='ram', help='image --cache ram/disk')
parser.add_argument('--device', type=str, default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
parser.add_argument('--workers', type=int, default=2, help='max dataloader workers (per RANK in DDP mode)')
parser.add_argument('--project', type=str, default='runs/train/testRIR/NEU-DE1', help='save to project/name')
parser.add_argument('--name', type=str, default='NEU-DE1-YOLOv8n_no_pre-RIR', help='save to project/name')
parser.add_argument('--resume', type=str, default='', help='resume training from last checkpoint')
parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'Adamw', 'NAdam', 'Radam', 'Adamw', 'RMSProp'], default='SGD')
parser.add_argument('--close_mosaic', type=int, default=0, help='(int) disable mosaic augmentation for final epochs')
parser.add_argument('--info', action='store_true', help='model info verbose')

parser.add_argument('--save', type=str2bool, default='True', help='save train checkpoints and predict results')
parser.add_argument('--save-period', type=int, default=1, help='Save checkpoint every x epochs (disabled if < 1)')
parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
parser.add_argument('--seed', type=int, default=0, help='Global training seed')
parser.add_argument('--deterministic', action='store_true', default=True, help='whether to enable deterministic mode')
parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
parser.add_argument('--rect', action='store_true', help='rectangular training')
parser.add_argument('--cos-lr', action='store_true', help='cosine LR scheduler')
parser.add_argument('--fraction', type=float, default=1.0, help='dataset fraction to train on (default is 1.0, all images in train set)')
parser.add_argument('--profile', action='store_true', help='profile ONNX and TensorRT speeds during training for 1000s')
parser.add_argument('--use_rir', action='store_true', default=True, help='RIR: random_interpolation_resize')

# Segmentation
parser.add_argument('--overlap_mask', type=str2bool, default='True', help='masks should overlap during training (segment train only)')
parser.add_argument('--mask_ratio', type=int, default=4, help='mask downsample ratio (segment train only)')

# Classification
parser.add_argument('--dropout', type=float, default=0.0, help='use dropout regularization (classify train only)')

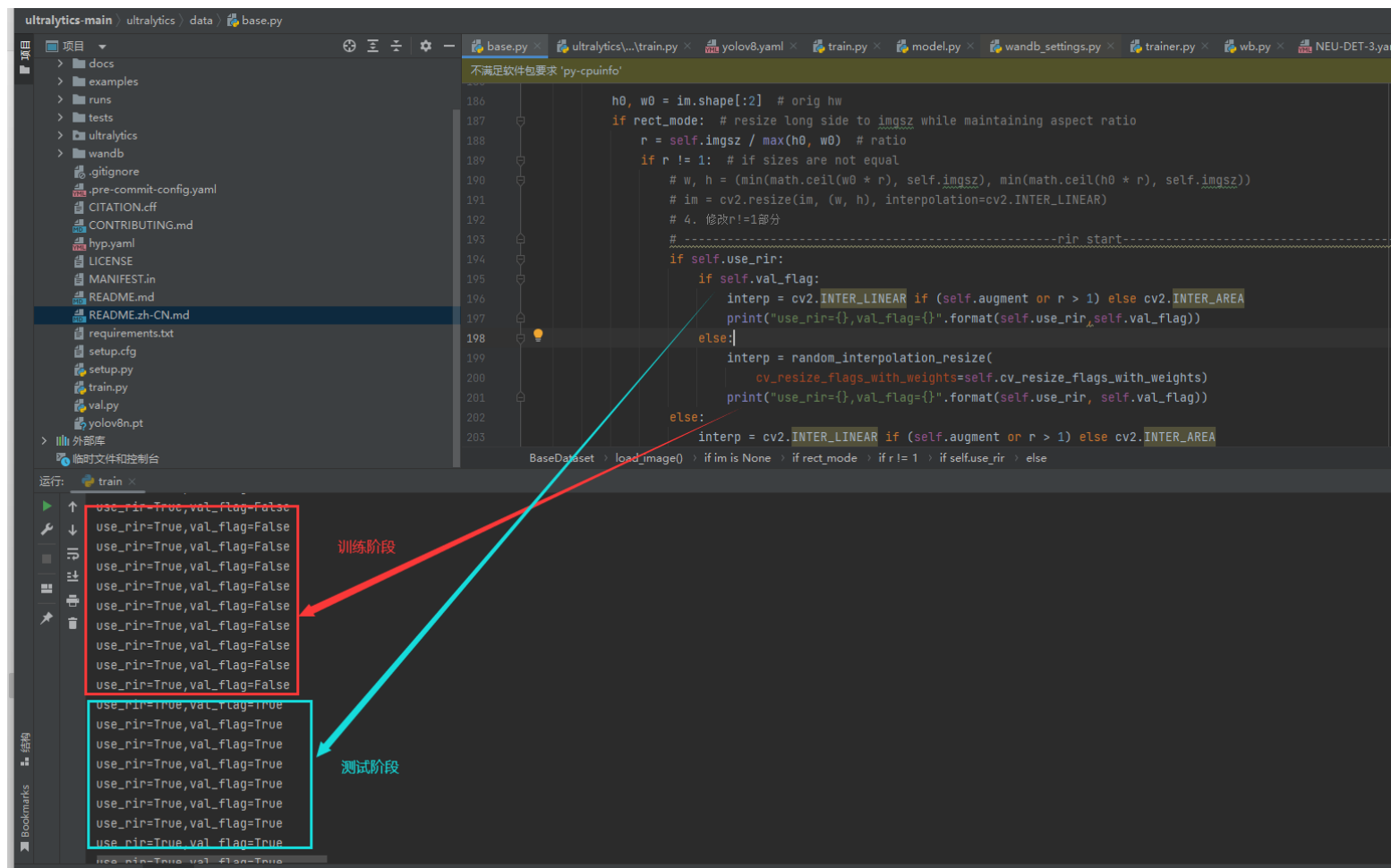
return parser.parse_known_args()[0]

class YOLOv8(YOLO):
    """
    """
    yaml_model_yaml_path
    weight_pretrained_model_path
```

3.1 YOLOv8 添加步骤

6.验证

- (1) 按照正常YOLOv8的训练步骤进行模型训练
- (2) 如果正确显示图中的内容,则表示添加成功,注释掉print,按原本的步骤运行即可



4.延伸创新点：（未做试验，欢迎继续探讨）

- (1) 分patch，每个patch采用不同的插值方式
- (2) 在训练和测试阶段均采用随机的插值方式
- (3) 在训练阶段最后 $n(n=30)$ 个epoch
- (4) 其他可以类比的方法也可以采用 random，试试效果

加工作量：

- (1) 搭配超参数进化，提升工作量（已在YOLOv5-6.1版本添加）
- (2) 搭配其他数据增强方法，组合成特定数据集（比如NEU-DET、GC10等）或者特定领域（工业检测、遥感领域等）的方法