



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

E-mail addresses: wandahang@mail.hfut.edu.cn (D. Wan)

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）

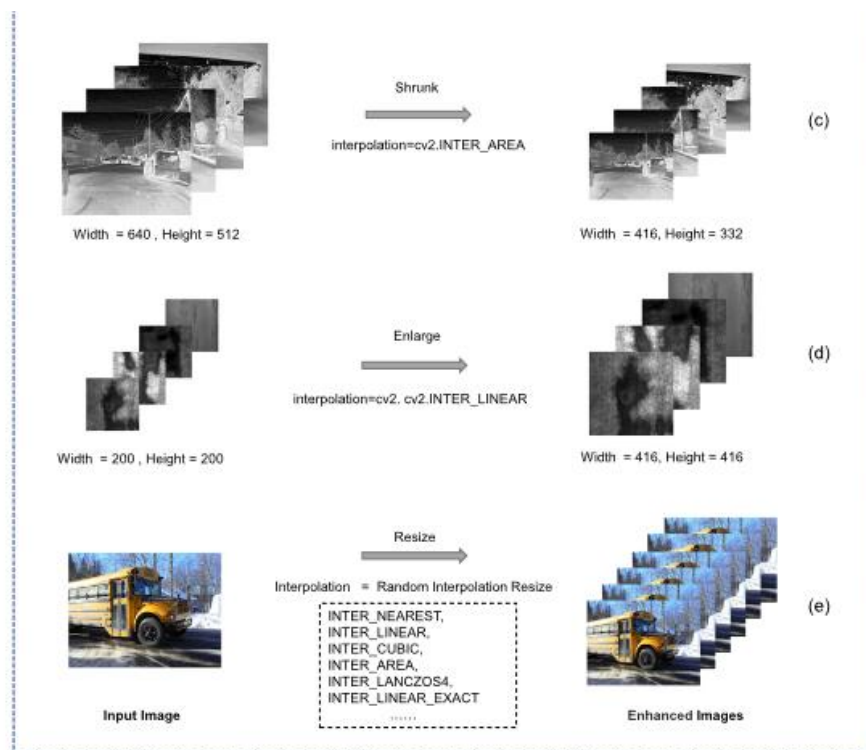


Fig. 2. Comparison of different methods of resize.

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

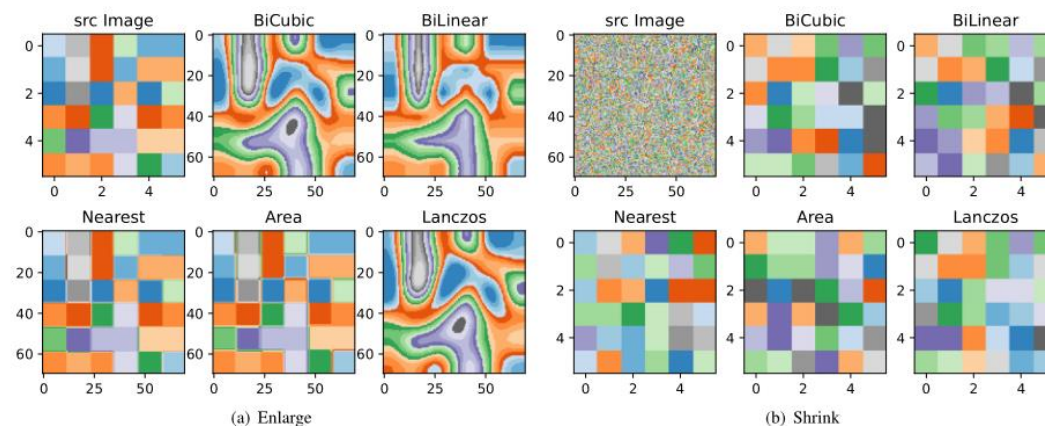


Fig. 3. Comparison of the results of various interpolation methods.

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(已完成)

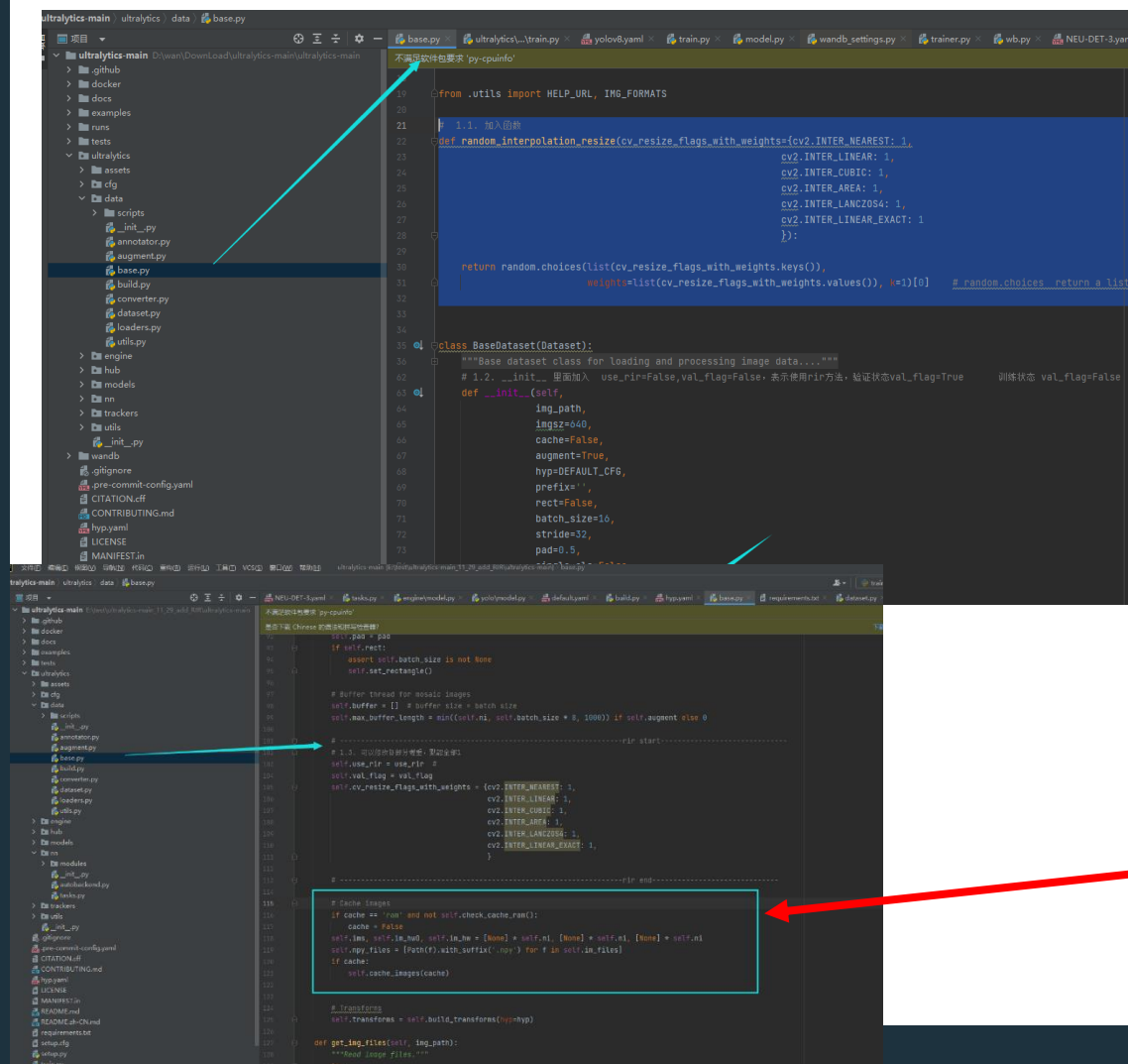
3.4 YOLOv5超参数进化 添加步骤(TODO)

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-----

Rir初始化的部分一定要放在cache的前面

3.1 YOLOv8 添加步骤

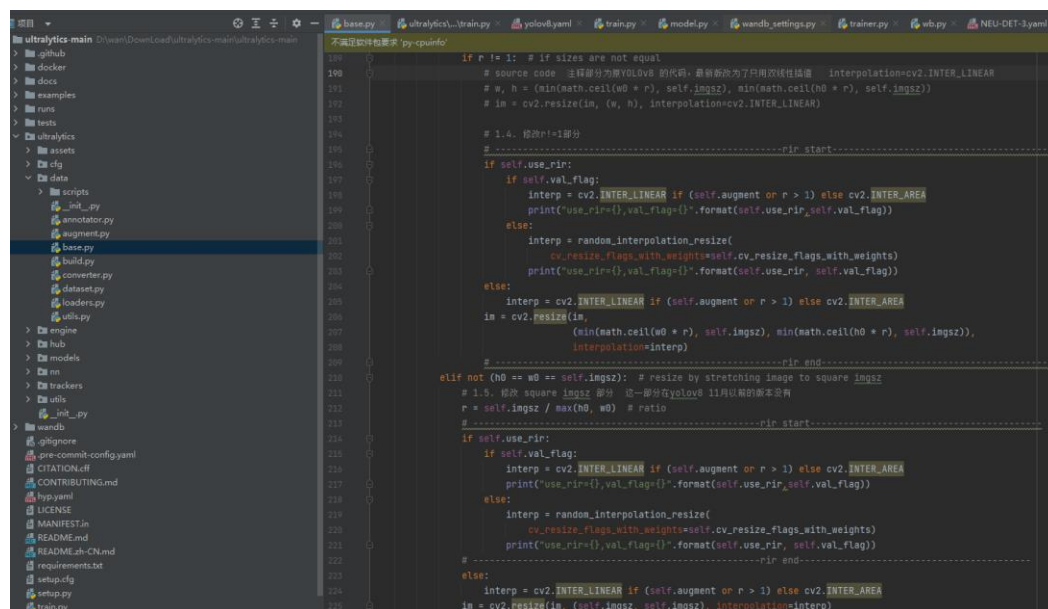
1. base.py 内部更改(11月以后)

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):
```



```
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)
```

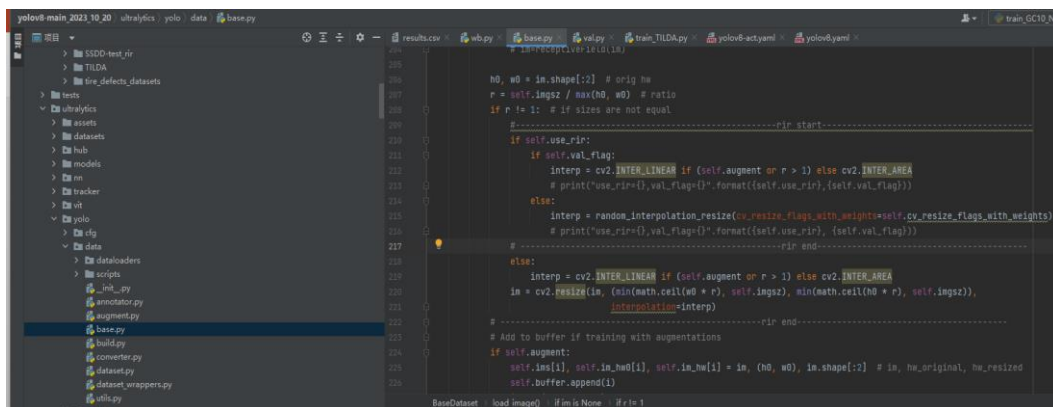
3.1 YOLOv8 添加步骤

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)

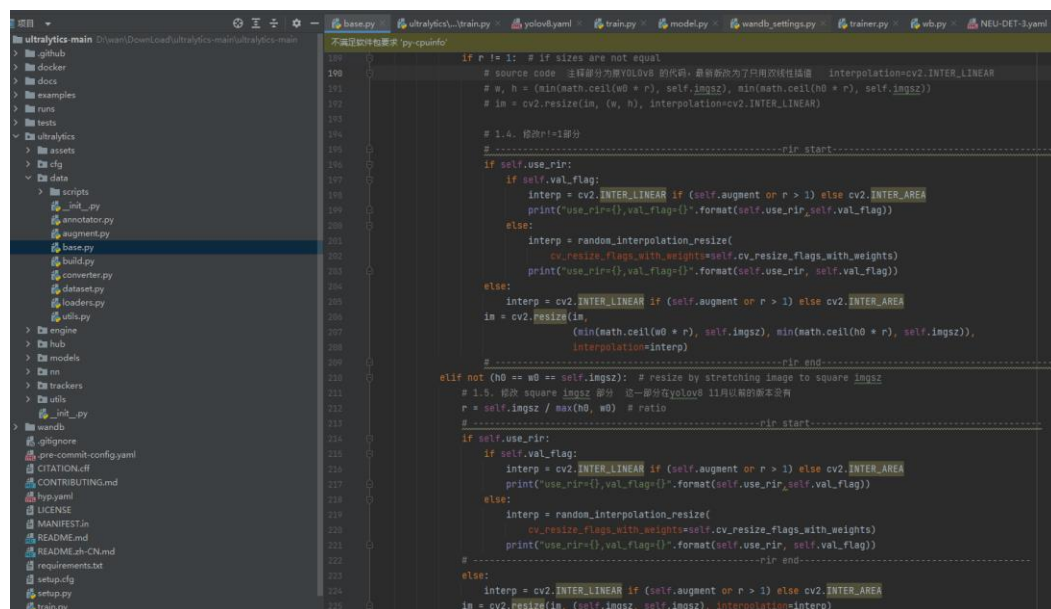
```
if rect_mode:
```

```
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-----

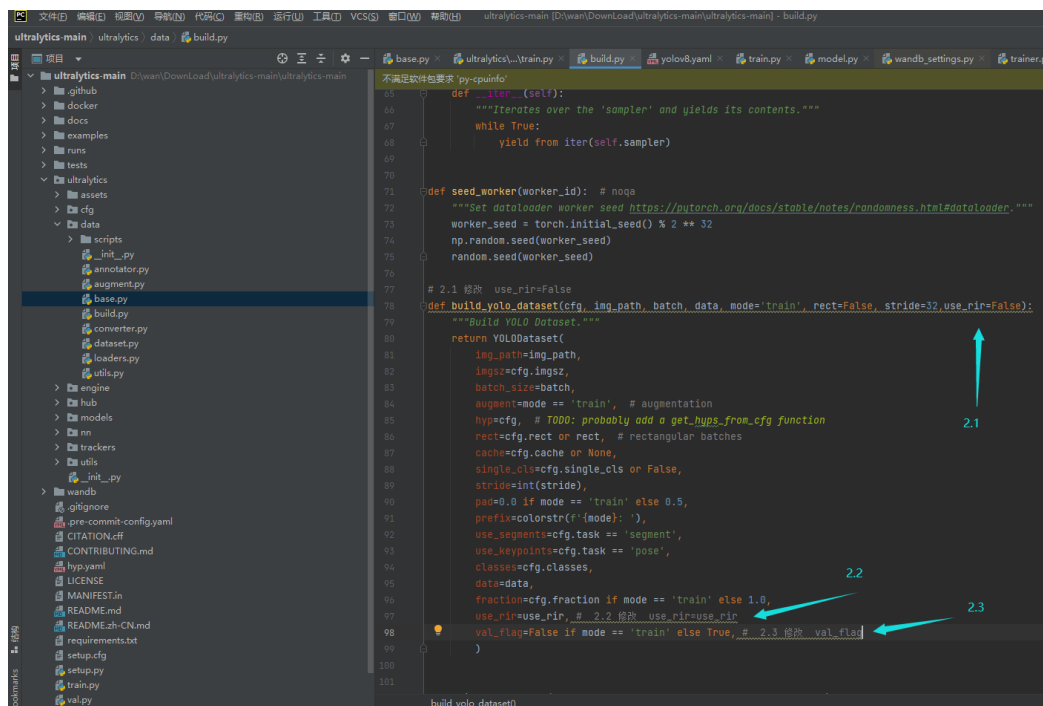
```
interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
im = cv2.resize(im, (self.imgsz, self.imgsz), interpolation=interp)
```


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  
    )
```

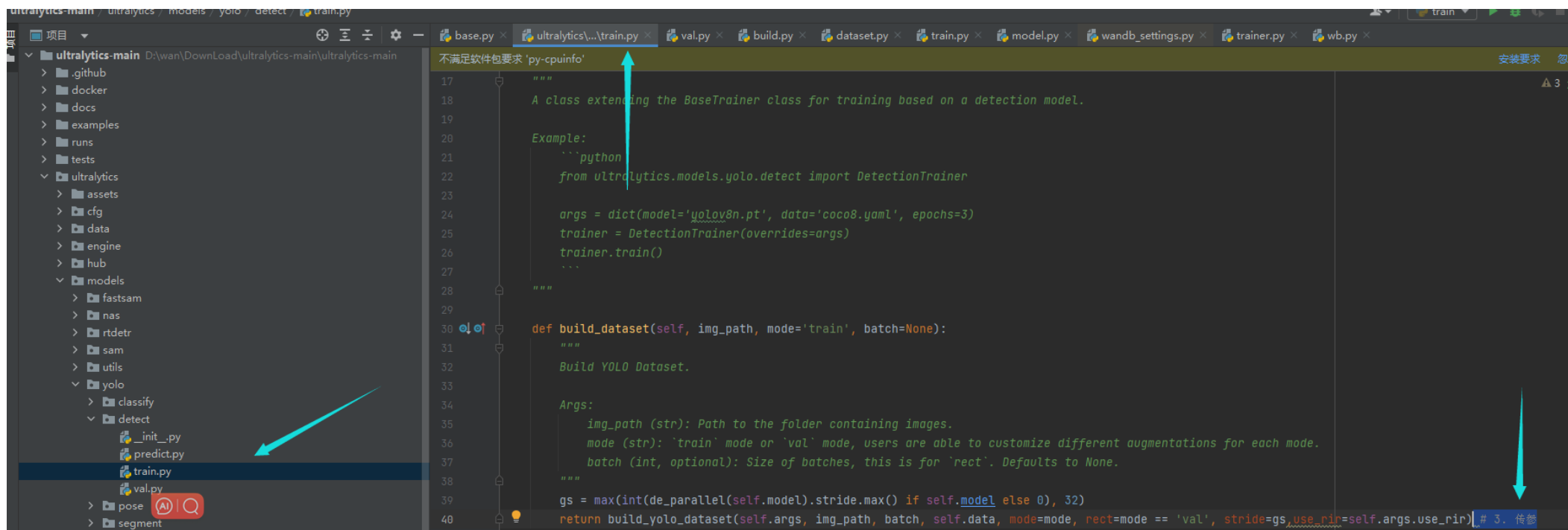
3.1 YOLOv8 添加步骤

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
```

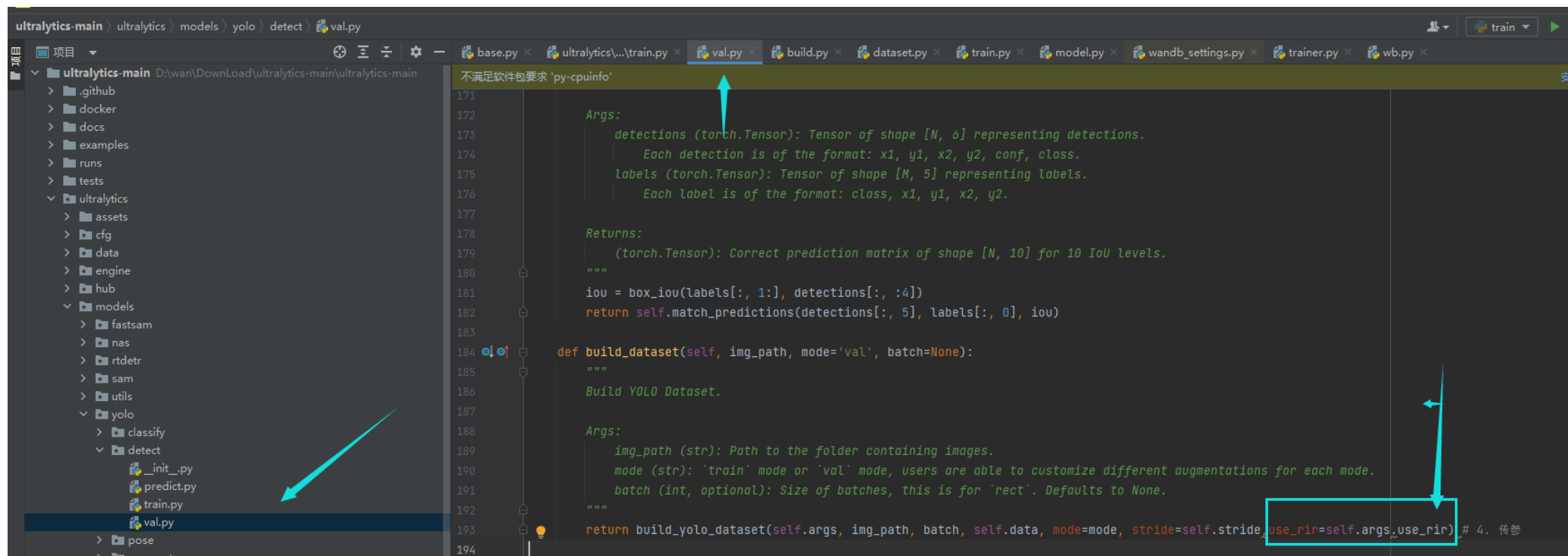


3.1 YOLOv8 添加步骤

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
```

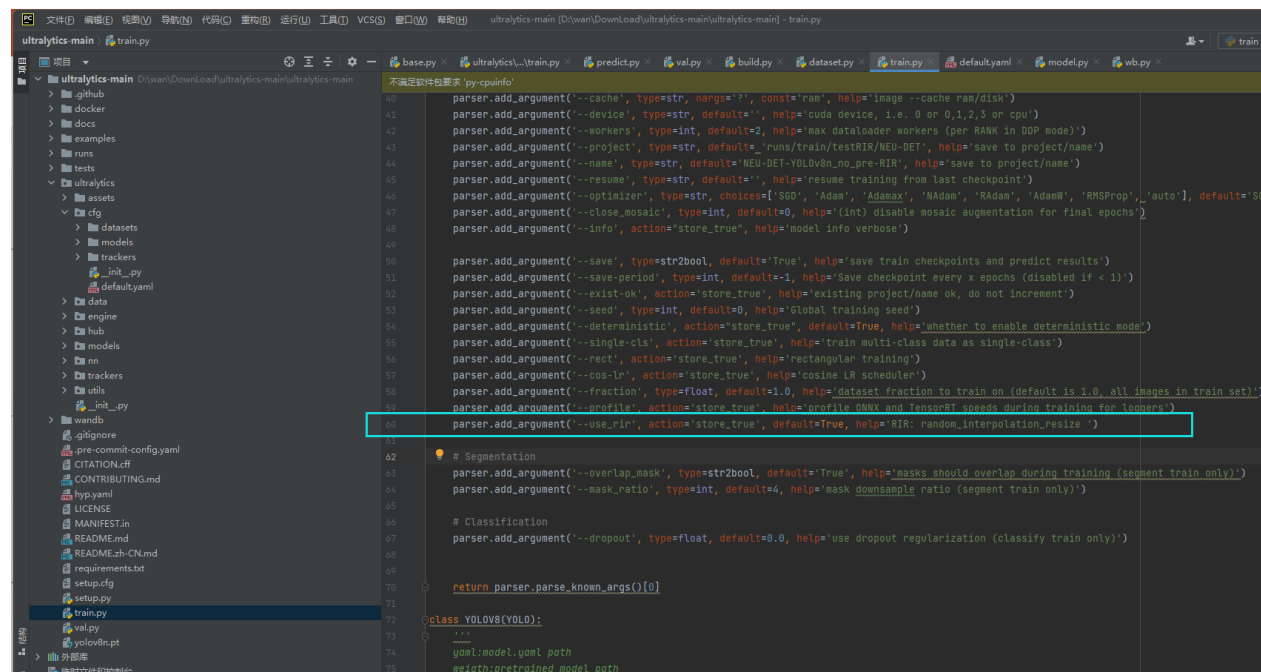
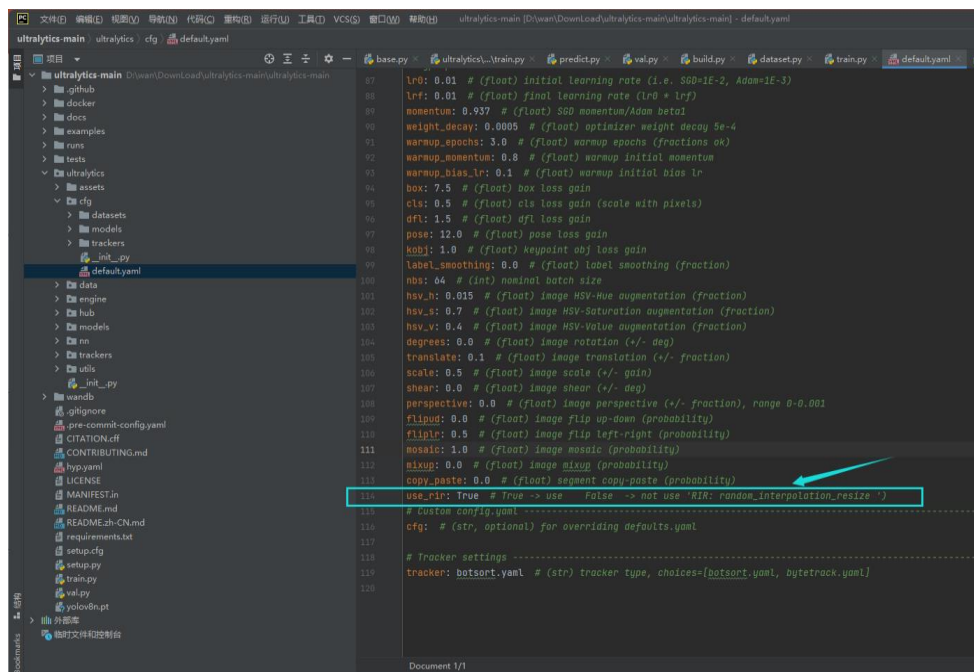


3.1 YOLOv8 添加步骤

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')
```

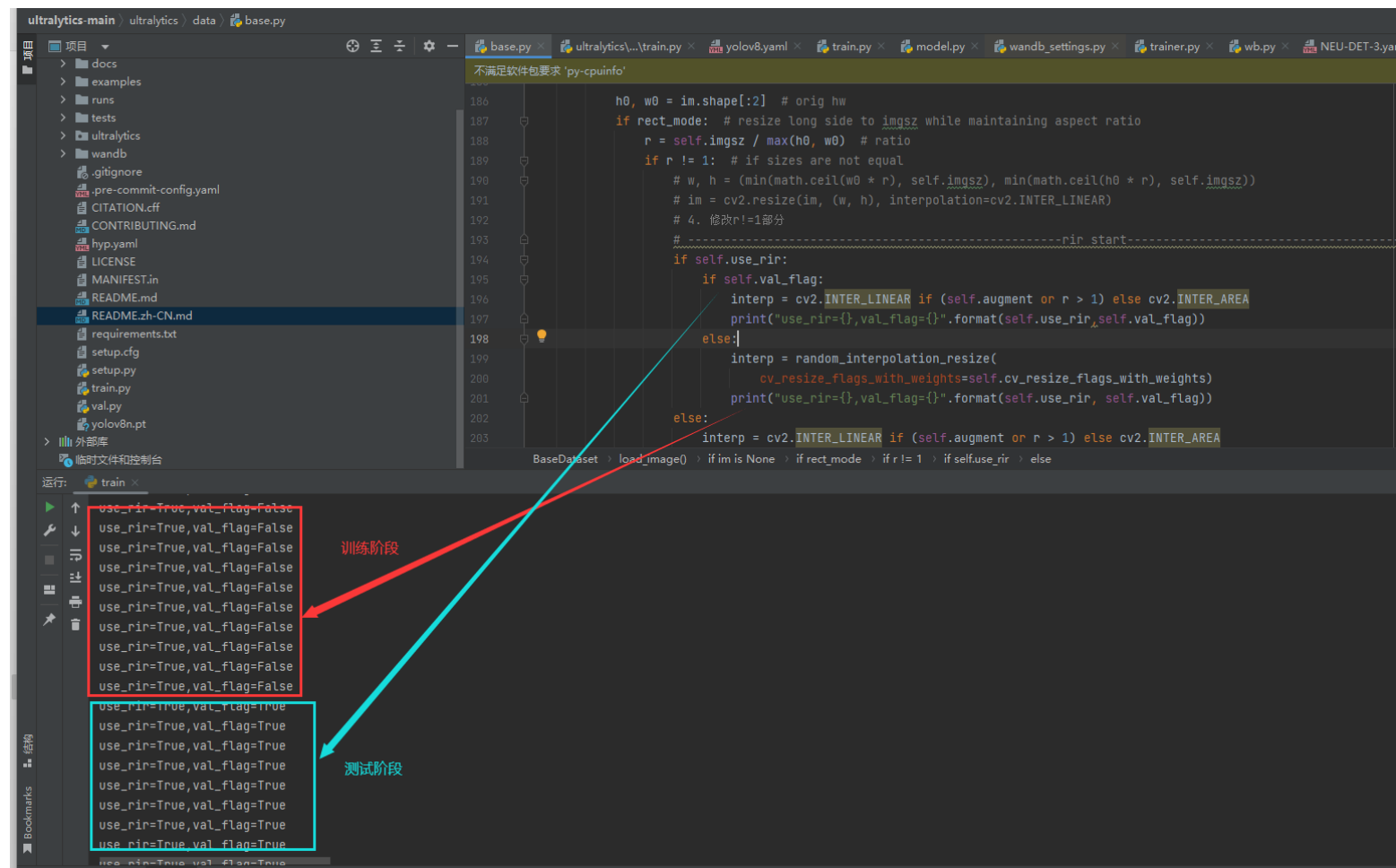
```
parser.add_argument('--use_rir', action='store_true', default=True, help='RIR: random_interpolation_resize')
```



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



3.3 YOLOv7 添加步骤

1.utils/dataset.py

- (1) 添加 random_interpolation_resize 函数
- (2) 更改load_image函数内的插值方式
- (3) 在 LoadImagesAndLabels 内添加use_rir和 val_flag __init__
- (4) 在 create_dataloader 内添加use_rir和 val_flag

2.train.py

- (5) 在 两处create_dataloader 内传入use_rir和 val_flag
- (6) 添加 顶层的 use_rir 参数

YOLOv7 项目地址（采用最新版进行演示，2023.12.19）

<https://github.com/WongKinYiu/yolov7>

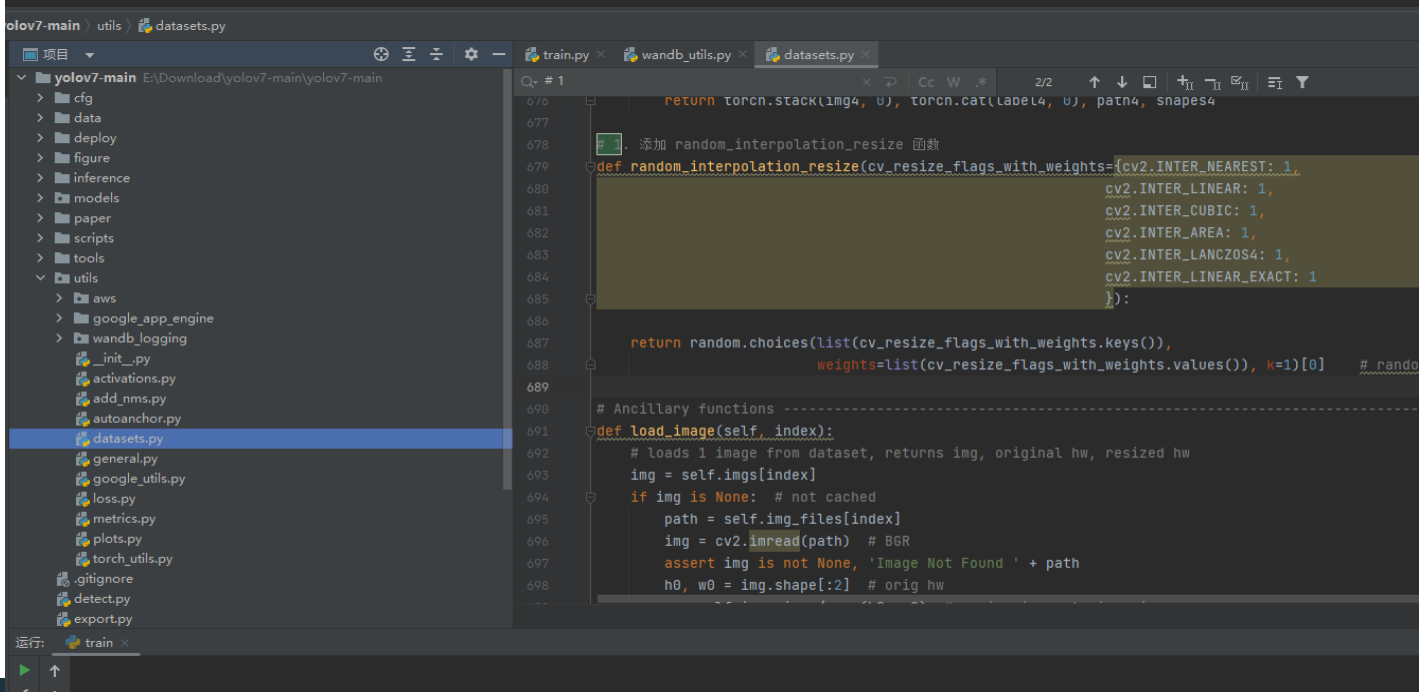
3.3 YOLOv7 添加步骤

1.utils/dataset.py (1) 添加 random_interpolation_resize 函数

1. 添加 random_interpolation_resize 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
```



3.3 YOLOv7 添加步骤

1.utils/dataset.py (2) 更改load_image函数内的插值方式

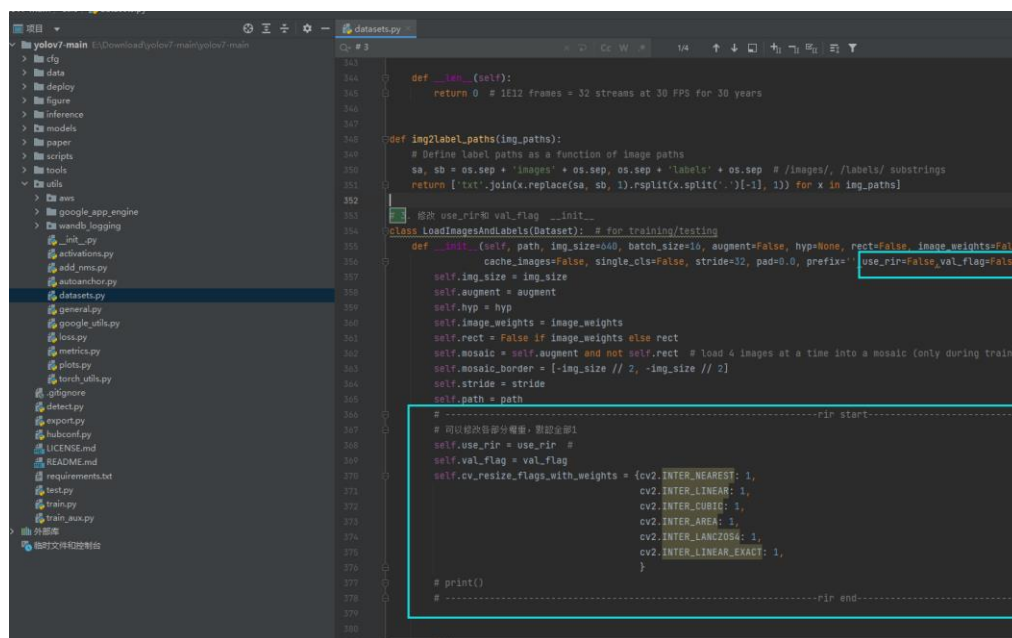
```
683         cv2.INTER_LINEAR_EXACT: 1,
684         cv2.INTER_LINEAR_EXACT: 1
685     }):
686
687     return random.choices(list(cv_resize_flags_with_weights.keys()),
688                           weights=list(cv_resize_flags_with_weights.values()), k=1)[0] # random.choices_return a l
689
690 # Ancillary functions -----
691 def load_image(self, index):
692     # loads 1 image from dataset, returns img, original hw, resized hw
693     img = self.imgs[index]
694     if img is None: # not cached
695         path = self.img_files[index]
696         img = cv2.imread(path) # BGR
697         assert img is not None, 'Image Not Found ' + path
698         h0, w0 = img.shape[:2] # orig hw
699         r = self.img_size / max(h0, w0) # resize image to img_size
700         if r != 1: # always resize down, only resize up if training with augmentation
701             # interp = cv2.INTER_AREA if r < 1 and not self.augment else cv2.INTER_LINEAR
702             # 2. 更改插值方式
703             # -----rir start-----
704             if self.use_rir:
705                 if self.val_flag:
706                     interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
707                     print("use_rir={},val_flag={}".format({self.use_rir},{self.val_flag}))
708                     # logging.info(f'use_rir={self.use_rir} val_flag={self.val_flag}')
709                 else:
710                     interp = random_interpolation_resize(cv_resize_flags_with_weights=self.cv_resize_flags_with_weight
711                     print("use_rir={},val_flag={}".format({self.use_rir},{self.val_flag}))
712                     # logging.info(f'use_rir={self.use_rir} val_flag={self.val_flag}')
713             else:
714                 interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
715             # -----rir end-----
716         img = cv2.resize(img, (int(w0 * r), int(h0 * r)), interpolation=interp)
717         return img, (h0, w0), img.shape[:2] # img, hw_original, hw_resized
718     else:
719         return self.imgs[index], self.img_hw0[index], self.img_hw[index] # img, hw_original, hw_resized
```

```
# Ancillary functions -----
def load_image(self, index):
    # loads 1 image from dataset, returns img, original hw, resized hw
    img = self.imgs[index]
    if img is None: # not cached
        path = self.img_files[index]
        img = cv2.imread(path) # BGR
        assert img is not None, 'Image Not Found ' + path
        h0, w0 = img.shape[:2] # orig hw
        r = self.img_size / max(h0, w0) # resize image to img_size
        if r != 1: # always resize down, only resize up if training with augmentation
            # interp = cv2.INTER_AREA if r < 1 and not self.augment else cv2.INTER_LINEAR
            # 2. 更改插值方式
            # -----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}))
                    # logging.info(f'use_rir={self.use_rir} val_flag={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}))
                    # logging.info(f'use_rir={self.use_rir} val_flag={self.val_flag}')
            else:
                interp = cv2.INTER_LINEAR if (self.augment or r > 1) else cv2.INTER_AREA
            # -----rir end-----
        img = cv2.resize(img, (int(w0 * r), int(h0 * r)), interpolation=interp)
        return img, (h0, w0), img.shape[:2] # img, hw_original, hw_resized
    else:
        return self.imgs[index], self.img_hw0[index], self.img_hw[index] # img, hw_original,
hw_resized
```

3.3 YOLOv7 添加步骤

1.utils/dataset.py

(3) 在 LoadImagesAndLabels 内添加 use_rir 和 val_flag __init__



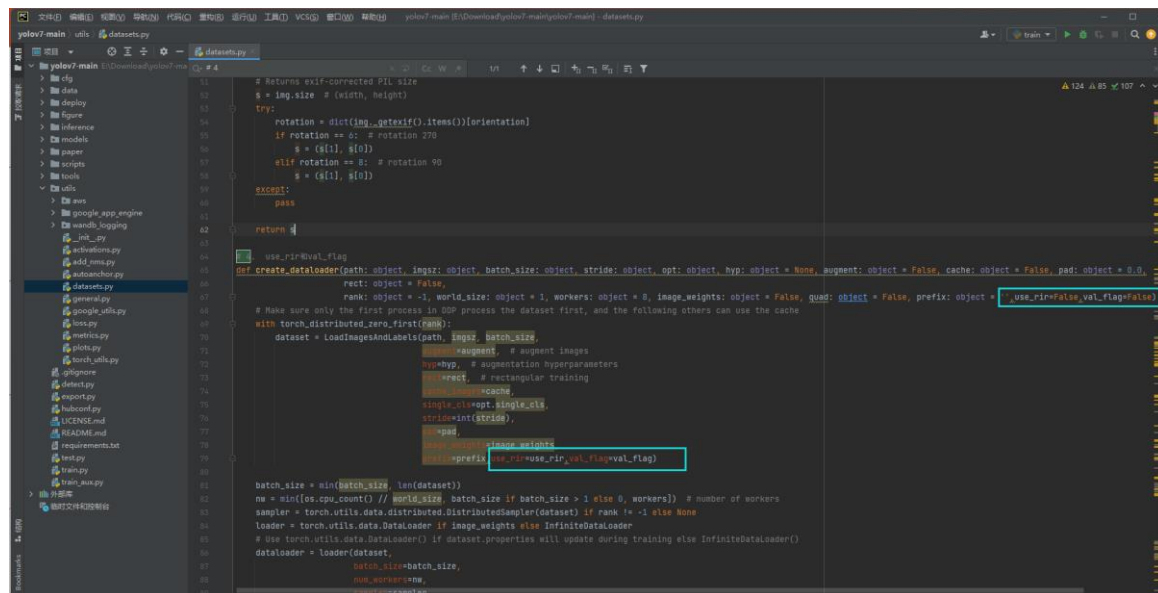
```
def __init__(self):  
    return 0 # 1E12 frames = 32 streams at 30 FPS for 30 years  
  
def img2label_paths(img_paths):  
    # Define label paths as a function of image paths  
    sa, sb = os.sep + 'images' + os.sep, os.sep + 'labels' + os.sep # /images/, /labels/ substrings  
    return ['%.join(x.replace(sa, sb, 1).rsplit(x.split('.')[0], 1)) for x in img_paths]  
  
class LoadImagesAndLabels(Dataset): # for training/testing  
    def __init__(self, path, img_size=640, batch_size=16, augment=False, hyp=None, rect=False, image_weights=False,  
        cache_images=False, single_cls=False, stride=32, pad=0.0, prefix=''): use_rir=False, val_flag=False  
        self.img_size = img_size  
        self.augment = augment  
        self.hyp = hyp  
        self.image_weights = image_weights  
        self.rect = False if image_weights else rect  
        self.mosaic = self.augment and not self.rect # load 4 images at a time into a mosaic (only during training)  
        self.mosaic_border = [-img_size // 2, -img_size // 2]  
        self.stride = stride  
        self.path = path  
  
        # -----rir start-----  
        # 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,  
        }  
  
        # print()  
        # -----rir end-----
```

```
# 3. 修改 use_rir和 val_flag __init__  
class LoadImagesAndLabels(Dataset): # for training/testing  
    def __init__(self, path, img_size=640, batch_size=16, augment=False, hyp=None, rect=False, image_weights=False,  
        cache_images=False, single_cls=False, stride=32, pad=0.0, prefix='', use_rir=False, val_flag=False):  
        self.img_size = img_size  
        self.augment = augment  
        self.hyp = hyp  
        self.image_weights = image_weights  
        self.rect = False if image_weights else rect  
        self.mosaic = self.augment and not self.rect # load 4 images at a time into a mosaic (only during training)  
        self.mosaic_border = [-img_size // 2, -img_size // 2]  
        self.stride = stride  
        self.path = path  
  
        # -----rir start-----  
        # 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,  
        }  
  
        # print()  
        # -----rir end-----
```

3.3 YOLOv7 添加步骤

1.utils/dataset.py

(4) 在 create_dataloader 内添加 use_rir 和 val_flag __init__



```
def create_dataloader(path: object, imgsiz: object, batch_size: object, stride: object, opt: object, hyp: object = None, augment: object = False, cache: object = False, pad: object = 0.0, rank: object = -1, world_size: object = 1, workers: object = 8, image_weights: object = False, quad: object = False, prefix: object = None, use_rir: bool = False, val_flag: bool = False):  
    # Make sure only the first process in DDP process the dataset first, and the following others can use the cache  
    with torch_distributed_zero_first(rank):  
        dataset = LoadImagesAndLabels(path, imgsiz, batch_size, augment, hyp, rect, single_cls, stride, pin_memory, use_rir, use_rir_val_flag=val_flag)  
    batch_size = min(batch_size, len(dataset))  
    nw = min([os.cpu_count() // world_size, batch_size if batch_size > 1 else 0, workers]) # number of workers  
    sampler = torch.utils.data.distributed.DistributedSampler(dataset) if rank != -1 else None  
    loader = torch.utils.data.DataLoader if image_weights else InfiniteDataLoader  
    # Use torch.utils.data.DataLoader() if dataset.properties will update during training else InfiniteDataLoader()  
    dataloader = loader(dataset,  
        batch_size=batch_size,  
        num_workers=nw,  
        sampler=sampler,  
        pin_memory=True,  
        collate_fn=LoadImagesAndLabels.collate_fn4 if quad else LoadImagesAndLabels.collate_fn)  
    return dataloader, dataset
```

4. use_rir和val_flag

def create_dataloader(path: object, imgsiz: object, batch_size: object, stride: object, opt: object, hyp: object = None, augment: object = False, cache: object = False, pad: object = 0.0, rect: object = False, rank: object = -1, world_size: object = 1, workers: object = 8, image_weights: object = False, quad: object = False, prefix: object = None, use_rir: bool = False, val_flag: bool = False):

"use_rir=False, val_flag=False) -> object:

Make sure only the first process in DDP process the dataset first, and the following others can use the cache

with torch_distributed_zero_first(rank):

dataset = LoadImagesAndLabels(path, imgsiz, batch_size,

augment=augment, # augment images

hyp=hyp, # augmentation hyperparameters

rect=rect, # rectangular training

cache_images=cache,

single_cls=opt.single_cls,

stride=int(stride),

pad=pad,

image_weights=image_weights,

prefix=prefix, use_rir=use_rir, val_flag=val_flag)

batch_size = min(batch_size, len(dataset))

nw = min([os.cpu_count() // world_size, batch_size if batch_size > 1 else 0, workers]) # number of workers

sampler = torch.utils.data.distributed.DistributedSampler(dataset) if rank != -1 else None

loader = torch.utils.data.DataLoader if image_weights else InfiniteDataLoader

Use torch.utils.data.DataLoader() if dataset.properties will update during training else InfiniteDataLoader()

dataloader = loader(dataset,

batch_size=batch_size,

num_workers=nw,

sampler=sampler,

pin_memory=True,

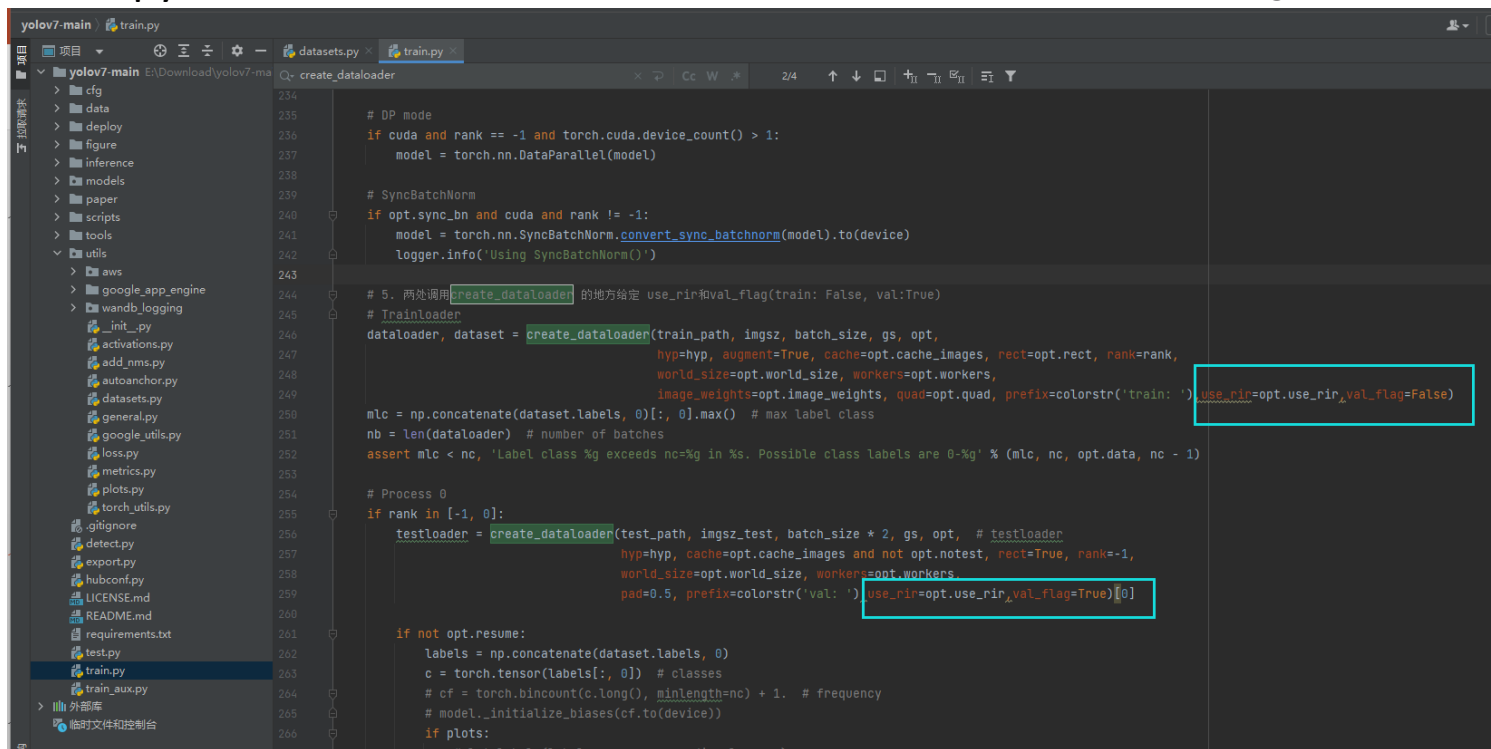
collate_fn=LoadImagesAndLabels.collate_fn4 if quad else LoadImagesAndLabels.collate_fn)

return dataloader, dataset

3.3 YOLOv7 添加步骤

2.train.py

(5) 在两处create_dataloader 内传入use_rir和 val_flag



```
234 # DP mode
235 if cuda and rank == -1 and torch.cuda.device_count() > 1:
236     model = torch.nn.DataParallel(model)
237
238 # SyncBatchNorm
239 if opt.sync_bn and cuda and rank != -1:
240     model = torch.nn.SyncBatchNorm.convert_sync_batchnorm(model).to(device)
241     logger.info('Using SyncBatchNorm()')
242
243 # 5. 两处调用create_dataloader 的地方传入 use_rir和val_flag(train: False, val:True)
244 # Trainloader
245 dataloader, dataset = create_dataloader(train_path, imgsz, batch_size, gs, opt,
246                                         hyp=hyp, augment=True, cache=opt.cache_images, rect=opt.rect, rank=rank,
247                                         world_size=opt.world_size, workers=opt.workers,
248                                         image_weights=opt.image_weights, quad=opt.quad, prefix=colorstr('train: '), use_rir=opt.use_rir, val_flag=False)
249
250 mlc = np.concatenate(dataset.labels, 0)[:, 0].max() # max label class
251 nb = len(dataloader) # number of batches
252 assert mlc < nc, 'Label class %g exceeds nc=%g in %s. Possible class labels are 0-%g' % (mlc, nc, opt.data, nc - 1)
253
254 # Process 0
255 if rank in [-1, 0]:
256     testloader = create_dataloader(test_path, imgsz_test, batch_size * 2, gs, opt, # testloader
257                                   hyp=hyp, cache=opt.cache_images and not opt.notest, rect=True, rank=-1,
258                                   world_size=opt.world_size, workers=opt.workers,
259                                   pad=0.5, prefix=colorstr('val: '), use_rir=opt.use_rir, val_flag=True)[0]
260
261 if not opt.resume:
262     labels = np.concatenate(dataset.labels, 0)
263     c = torch.tensor(labels[:, 0]) # classes
264     # cf = torch.bincount(c.long(), minlength=nc) + 1. # frequency
265     # model._initialize_biases(cf.to(device))
266     if plots:
```

```
# Trainloader
dataloader, dataset = create_dataloader(train_path, imgsz, batch_size, gs, opt,
                                        hyp=hyp, augment=True, cache=opt.cache_images, rect=opt.rect, rank=rank,
                                        world_size=opt.world_size, workers=opt.workers,
                                        image_weights=opt.image_weights, quad=opt.quad, prefix=colorstr('train: '), use_rir=opt.use_rir, val_flag=False)
```

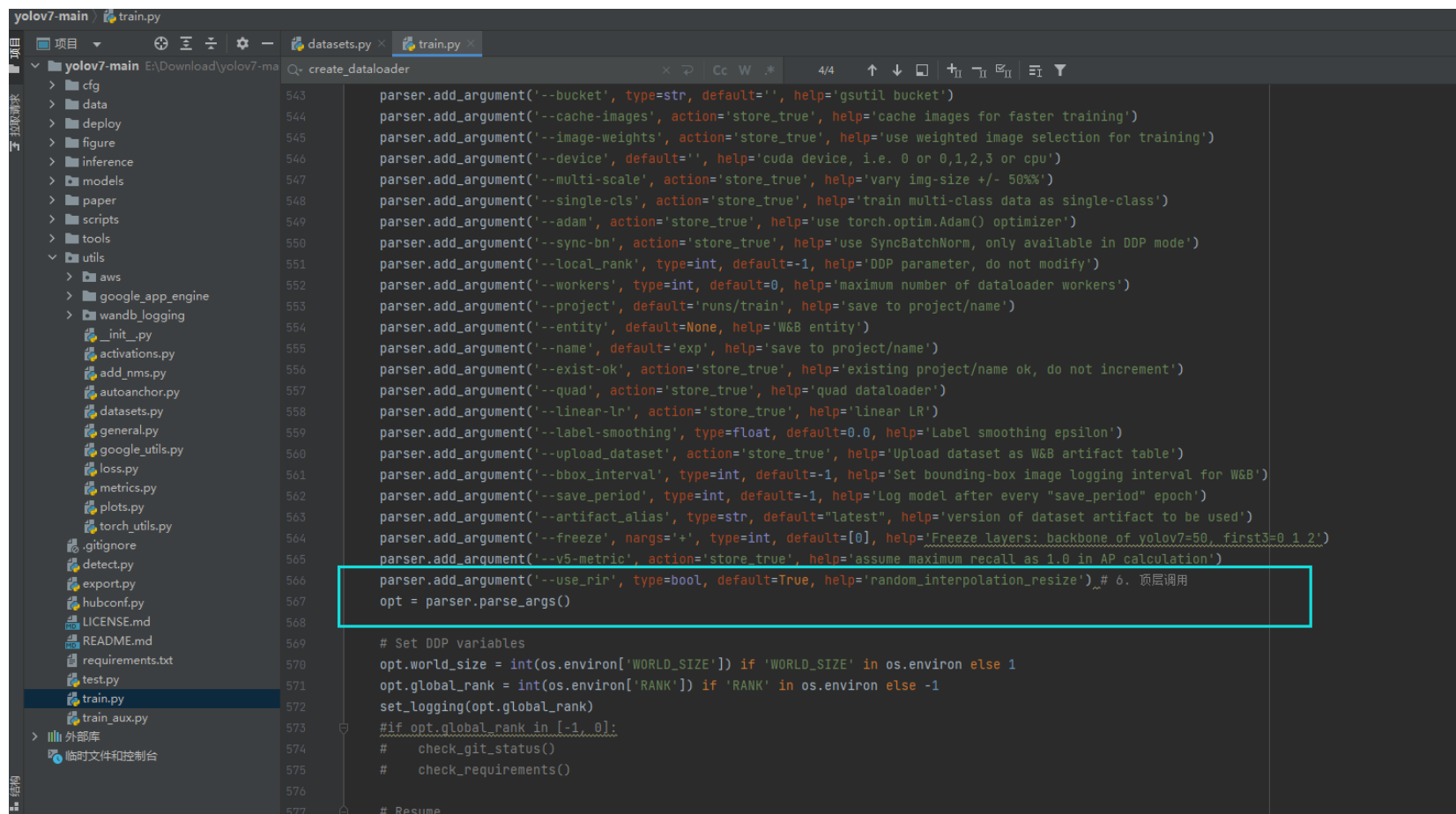
```
testloader = create_dataloader(test_path, imgsz_test, batch_size * 2, gs, opt, # testloader
                                hyp=hyp, cache=opt.cache_images and not opt.notest, rect=True, rank=-1,
                                world_size=opt.world_size, workers=opt.workers,
                                pad=0.5, prefix=colorstr('val: '), use_rir=opt.use_rir, val_flag=True)[0]
```

3.3 YOLOv7 添加步骤

2.train.py

(6) 添加 顶层的 use_rir 参数

`parser.add_argument('--use_rir', type=bool, default=True, help='random_interpolation_resize') # 6. 顶层调用`



```
543 parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
544 parser.add_argument('--cache-images', action='store_true', help='cache images for faster training')
545 parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
546 parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
547 parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%')
548 parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
549 parser.add_argument('--adam', action='store_true', help='use torch.optim.Adam() optimizer')
550 parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
551 parser.add_argument('--local_rank', type=int, default=-1, help='DDP parameter, do not modify')
552 parser.add_argument('--workers', type=int, default=0, help='maximum number of dataloader workers')
553 parser.add_argument('--project', default='runs/train', help='save to project/name')
554 parser.add_argument('--entity', default=None, help='W&B entity')
555 parser.add_argument('--name', default='exp', help='save to project/name')
556 parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
557 parser.add_argument('--quad', action='store_true', help='quad dataloader')
558 parser.add_argument('--linear-lr', action='store_true', help='linear LR')
559 parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')
560 parser.add_argument('--upload_dataset', action='store_true', help='Upload dataset as W&B artifact table')
561 parser.add_argument('--bbox_interval', type=int, default=-1, help='Set bounding-box image logging interval for W&B')
562 parser.add_argument('--save_period', type=int, default=-1, help='Log model after every "save_period" epoch')
563 parser.add_argument('--artifact_alias', type=str, default="latest", help='version of dataset artifact to be used')
564 parser.add_argument('--freeze', nargs='+', type=int, default=[0], help='Freeze layers: backbone of yolov7=50, first3=0 1 2')
565 parser.add_argument('--v5-metric', action='store_true', help='assume maximum recall as 1.0 in AP calculation')
566 parser.add_argument('--use_rir', type=bool, default=True, help='random_interpolation_resize') # 6. 顶层调用
567 opt = parser.parse_args()
568
569 # Set DDP variables
570 opt.world_size = int(os.environ['WORLD_SIZE']) if 'WORLD_SIZE' in os.environ else 1
571 opt.global_rank = int(os.environ['RANK']) if 'RANK' in os.environ else -1
572 set_logging(opt.global_rank)
573 if opt.global_rank in [-1, 0]:
574     # check_git_status()
575     # check_requirements()
576
577 # Resume
```

3.3 YOLOv7 添加步骤

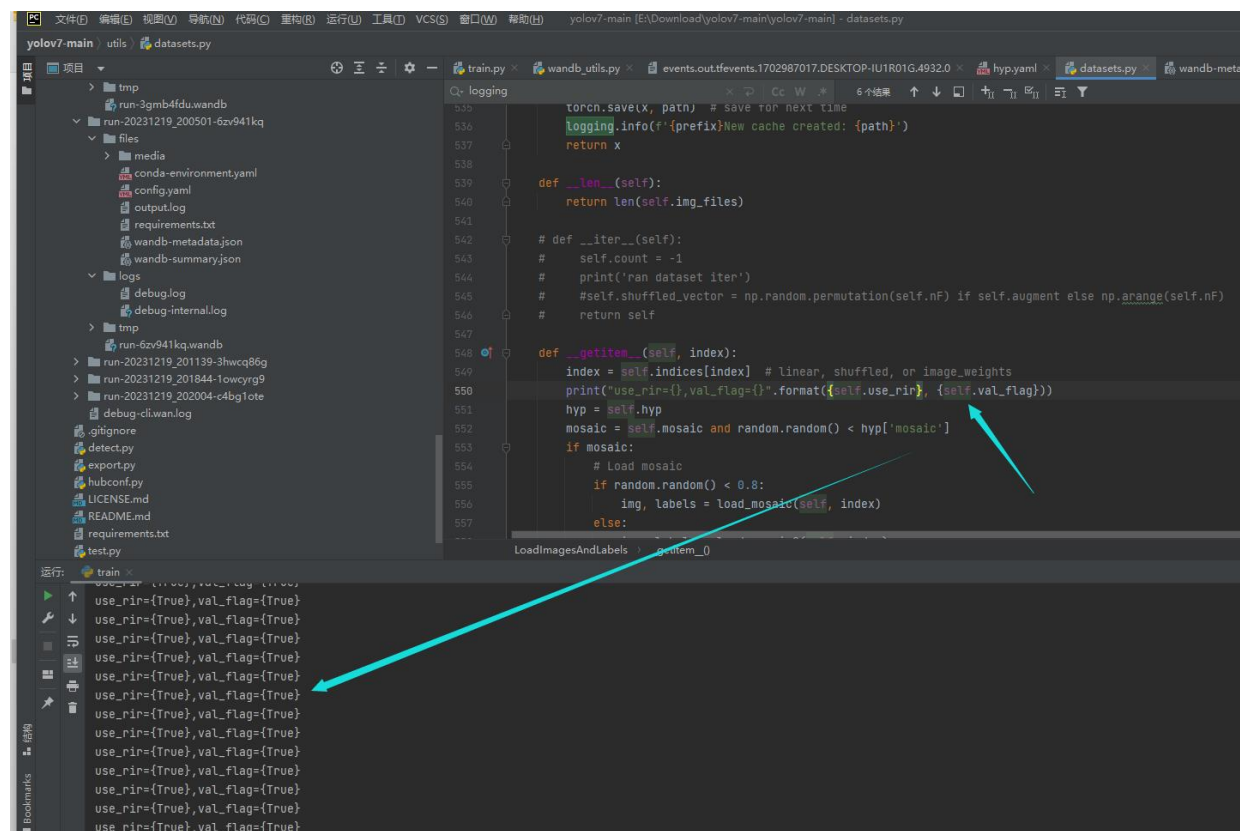
3.验证

- (1) 按照正常YOLOv7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

utils/dataset.py

LoadImagesAndLabels

__getitem__



```
def __getitem__(self, index):
    index = self.indices[index] # linear, shuffled, or image_weights
    print("use_rir={}, val_flag={}".format(self.use_rir, self.val_flag))
    hyp = self.hyp
    mosaic = self.mosaic and random.random() < hyp['mosaic']
    if mosaic:
        # Load mosaic
        if random.random() < 0.8:
            img, labels = load_mosaic(self, index)
        else:
            img, labels = load_mosaic9(self, index)
    else:
        img, labels = load_image(self, index)
    img = img.transpose(2, 0, 1) # HWC to CHW
    labels = labels.transpose(0, 1) # [class, xywh] to [xywh, class]
    img = img.to(self.device)
    labels = labels.to(self.device)

    return img, labels
```

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