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）

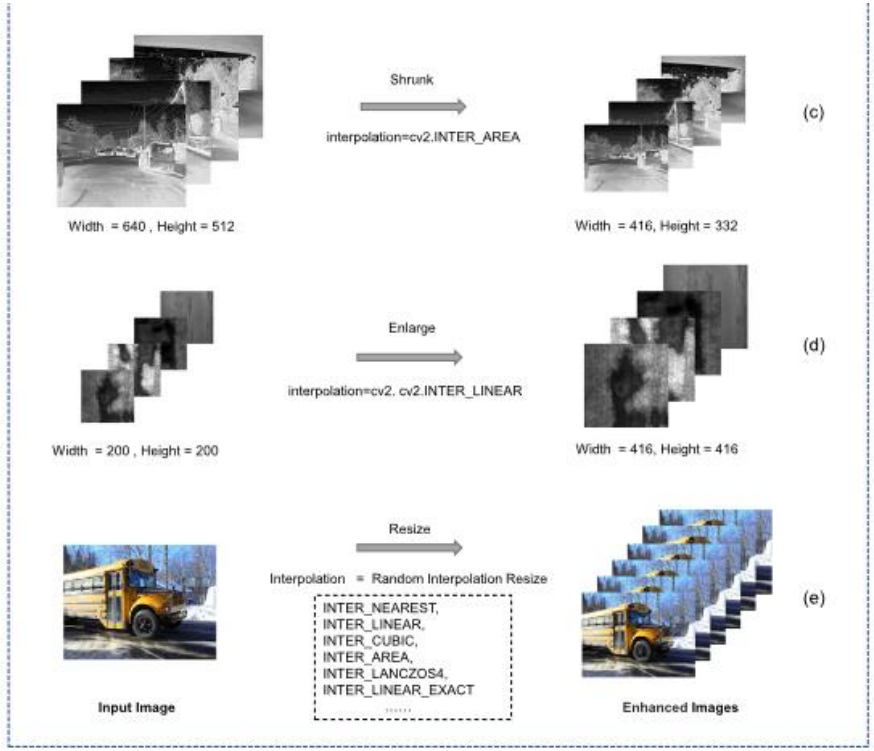


Fig. 2. Comparison of different methods of resize.

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

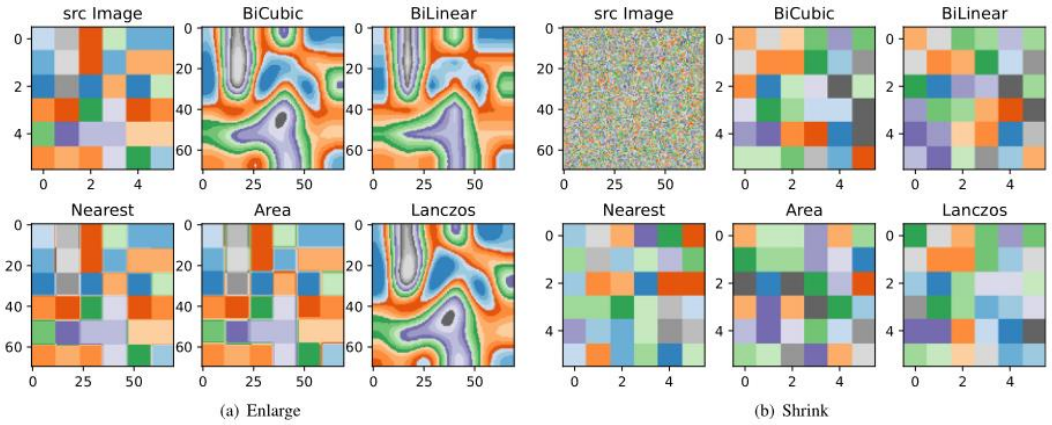


Fig. 3. Comparison of the results of various interpolation methods.

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

```
# ----------------------------------------------------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----------------------------------------
```

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
```

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)

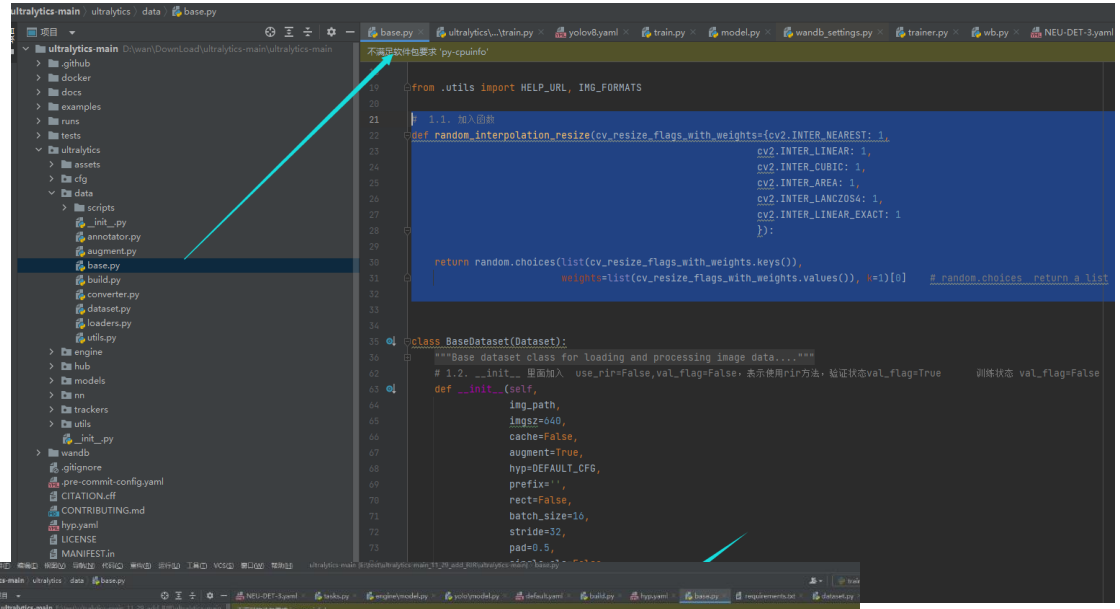
3.4 YOLOv5超参数进化 添加步骤(TODO)

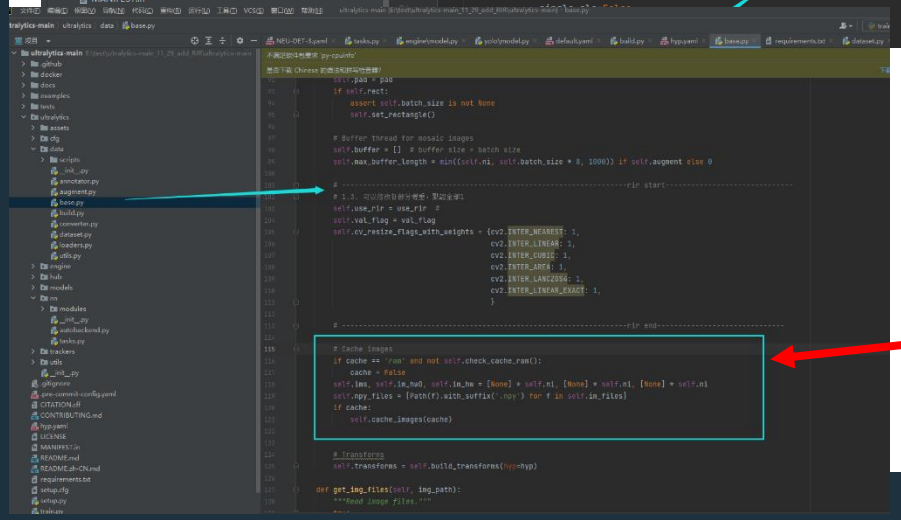
3.1 YOLOv8 添加步骤

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，可以修改各部分權重，默認全部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

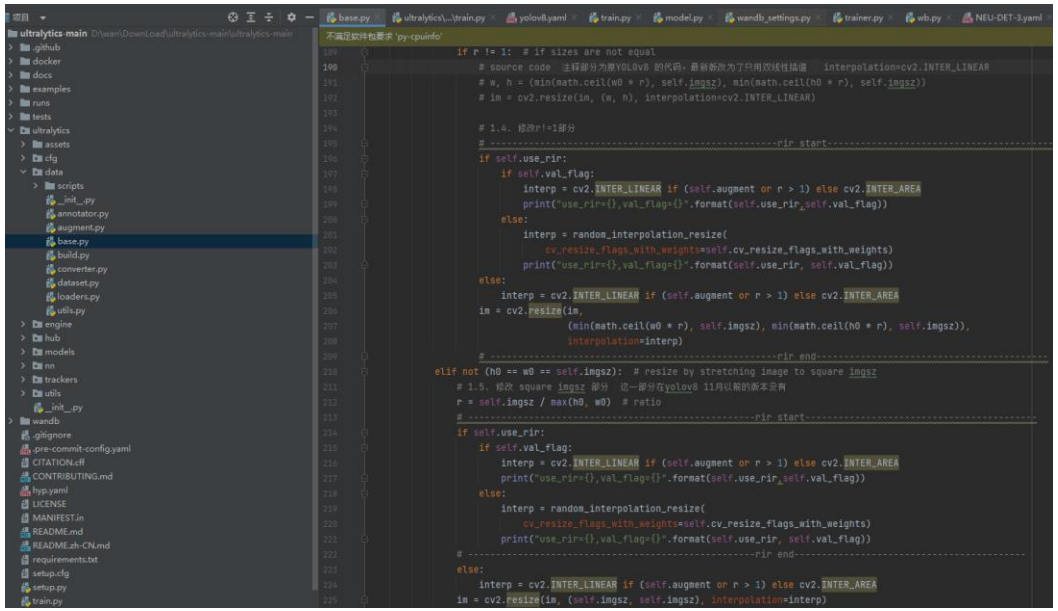
1. base.py 内部更改(11月以后)

1.4. 修改r!=1部分

1.5. 修改 square imgsz 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 imgsz 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 imgsz
    # 1.5. 修改 square imgsz 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))
    # ----------------------------------------------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)
```

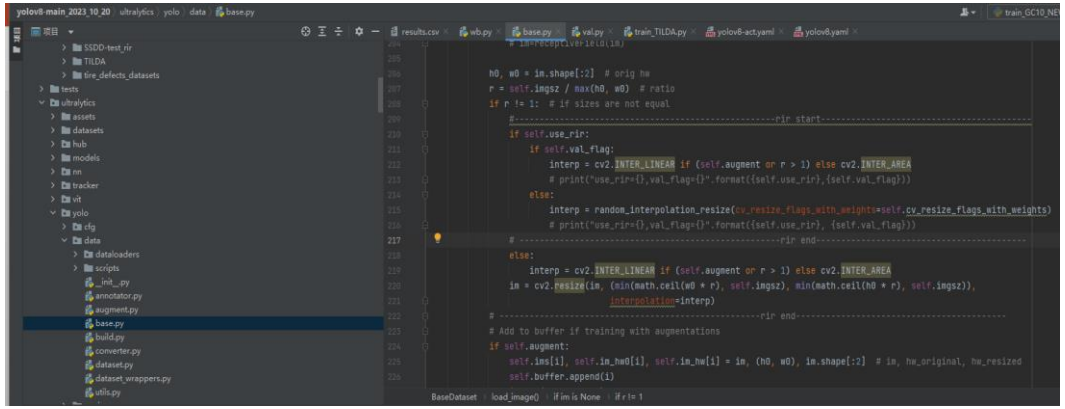
3.1 YOLOv8 添加步骤

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)

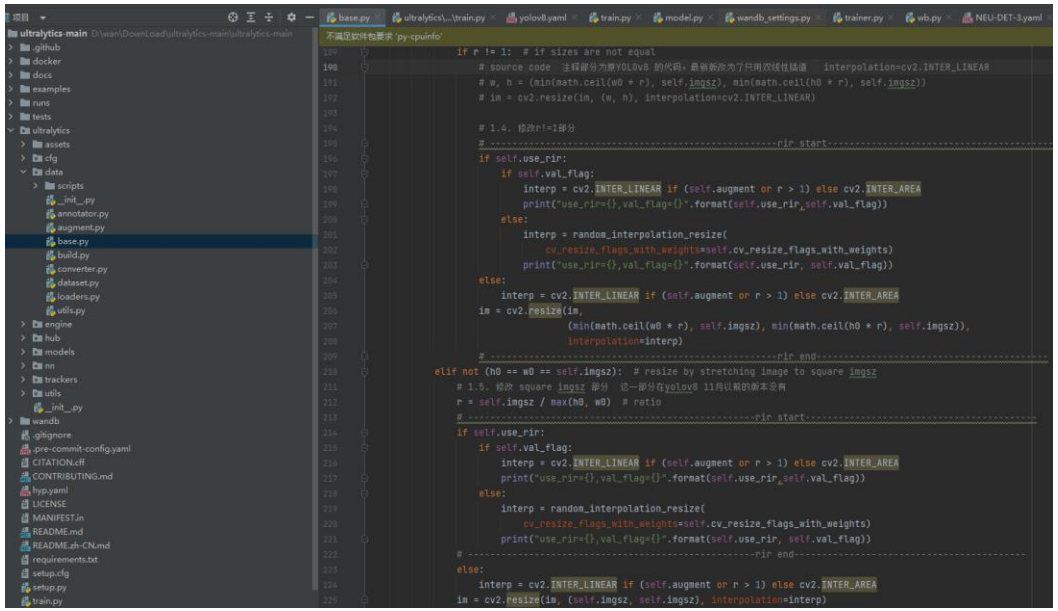
```
if rect_mode:
```

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

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

1.4. 修改r!=1部分

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



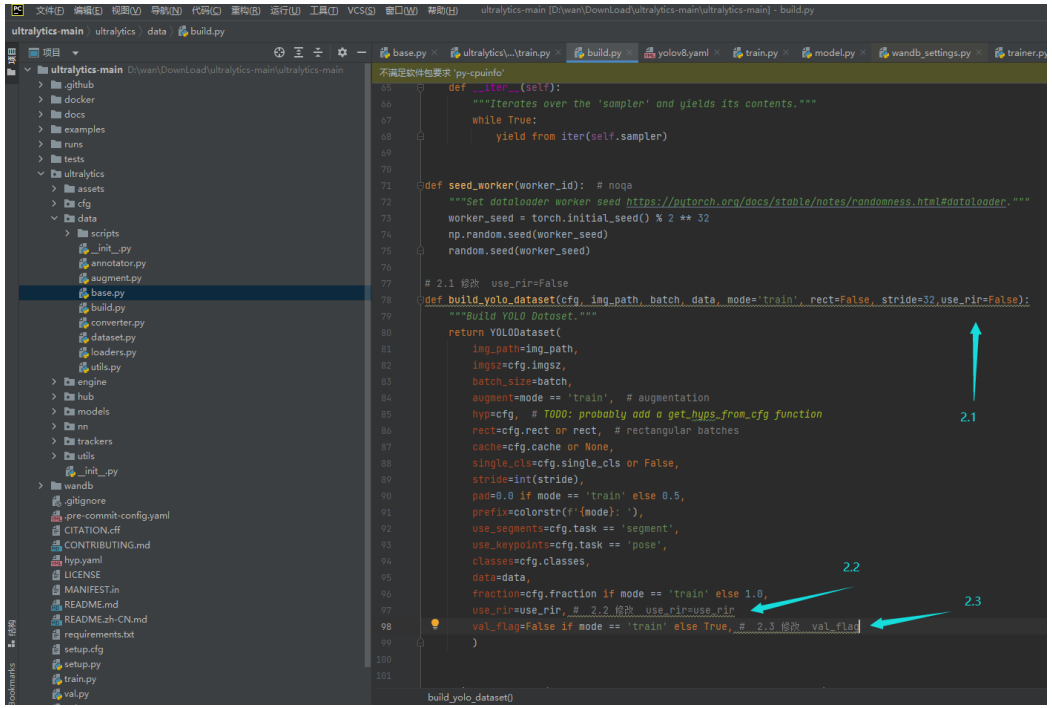
```
if rect_mode:  # resize long side to imgsz while maintaining aspect ratio
    r = self.imgsz / max(h0, w0)  # ratio
    if r != 1:  # if sizes are not equal
        # source code  注释部分为原YOLOv8 的代码，最新版改为了只用双线性插值
interpolation=cv2.INTER_LINEAR
        # w, h = (min(math.ceil(w0 * r), self.imgsz), min(math.ceil(h0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

ultralytics/data/build.py

ultralytics/yolo/data/build.py

2. build.py build_yolo_dataset




```python
# 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,
        imgsz=cfg.imgsz,
        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=colorstr(f'{mode}: '),
        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
        )
```

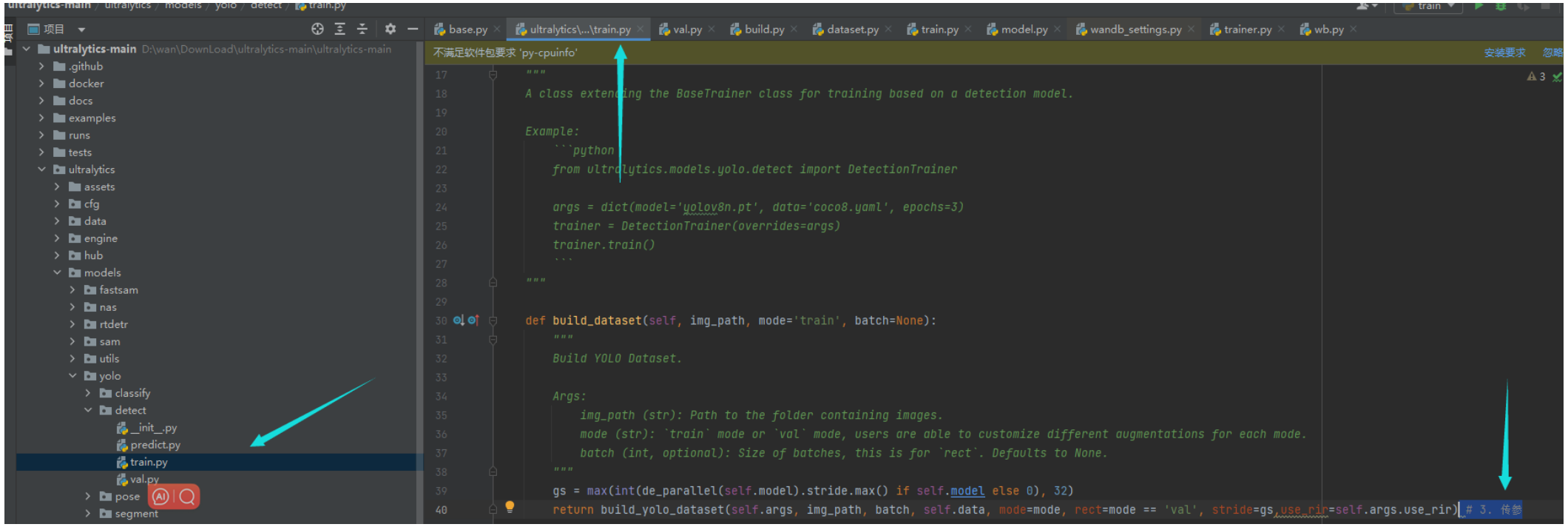
3.1 YOLOv8 添加步骤

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
```



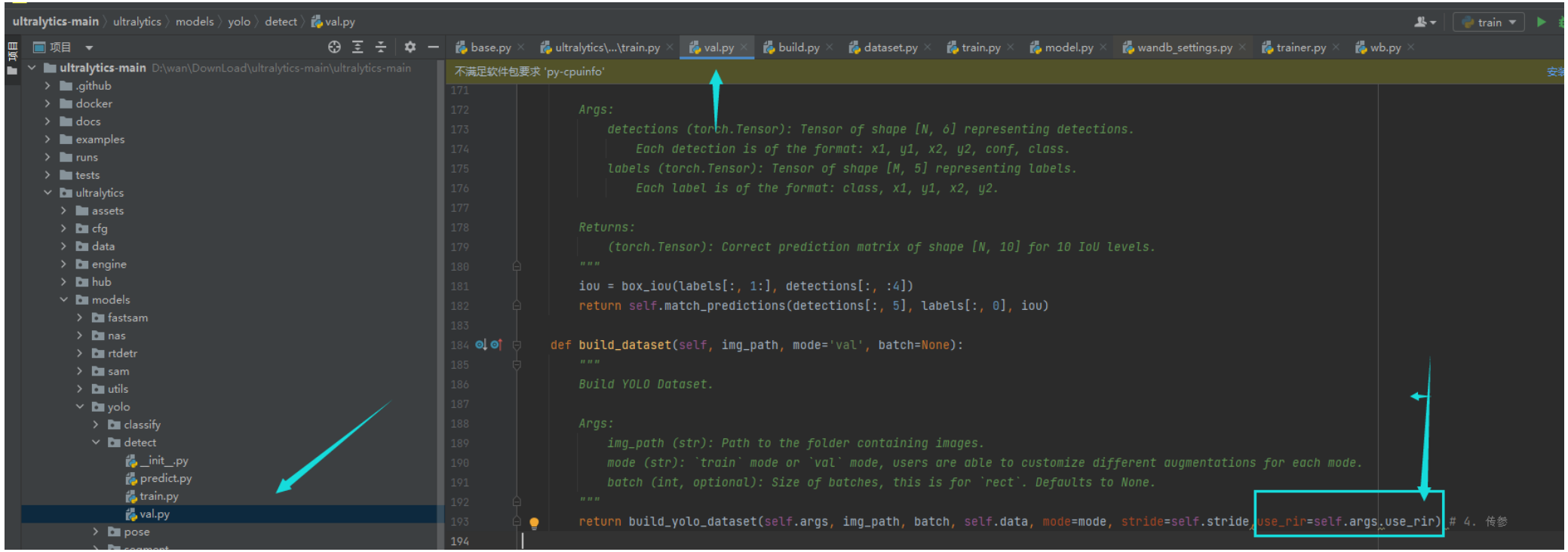
3.1 YOLOv8 添加步骤

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
```

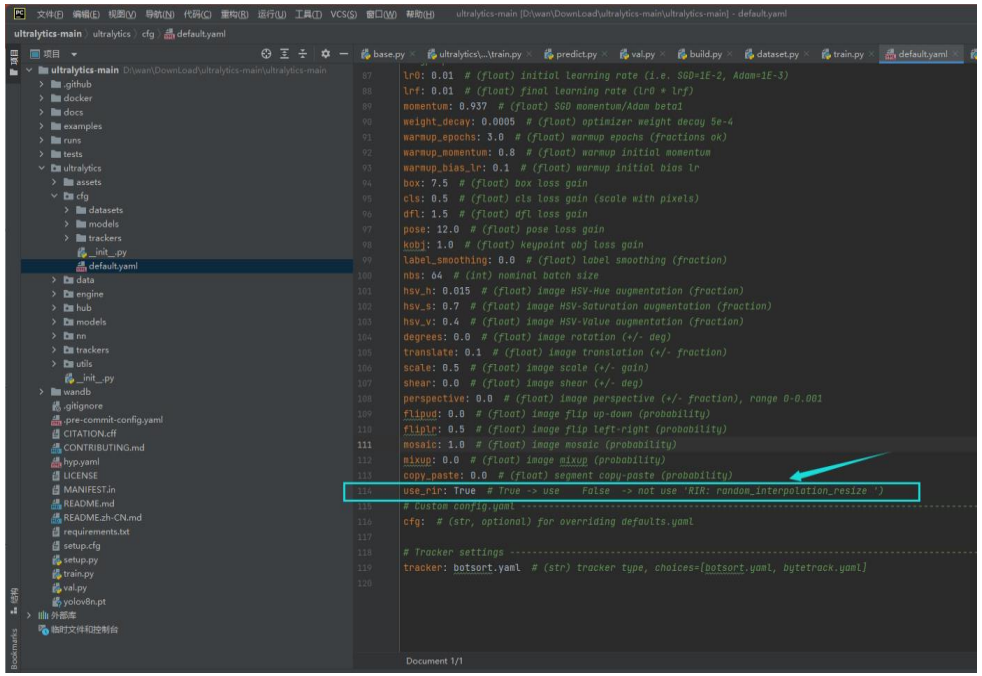


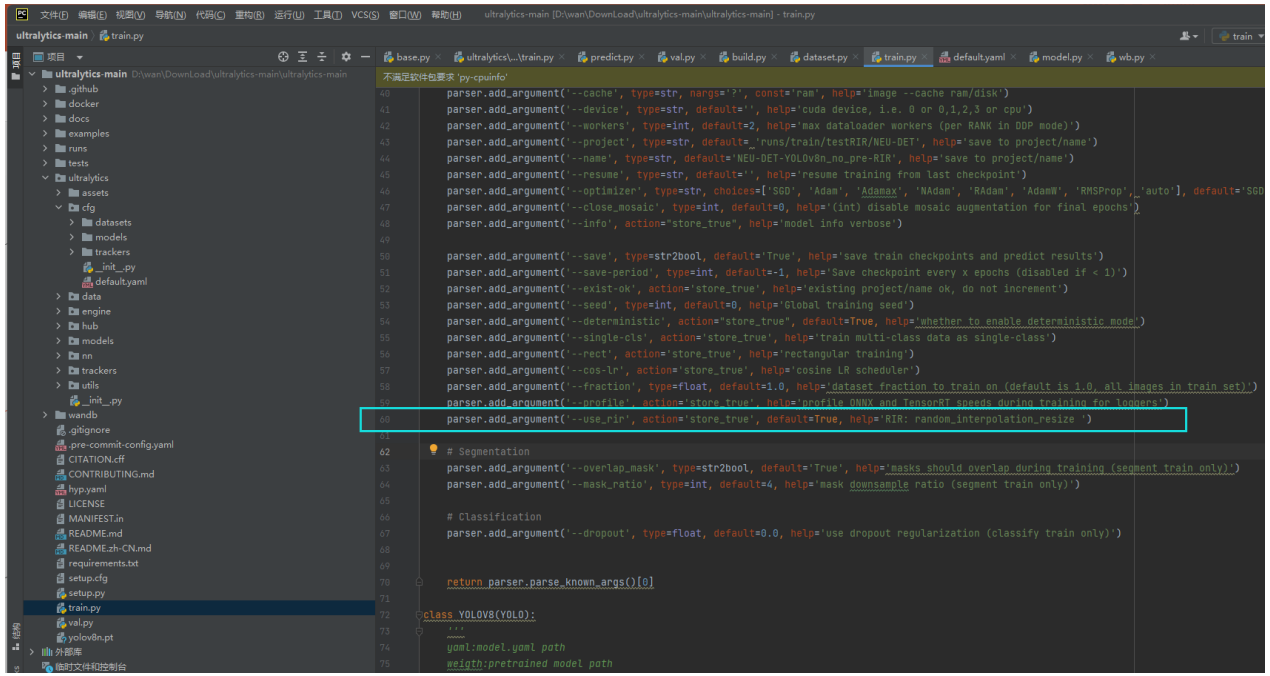
3.1 YOLOv8 添加步骤

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 ')
```

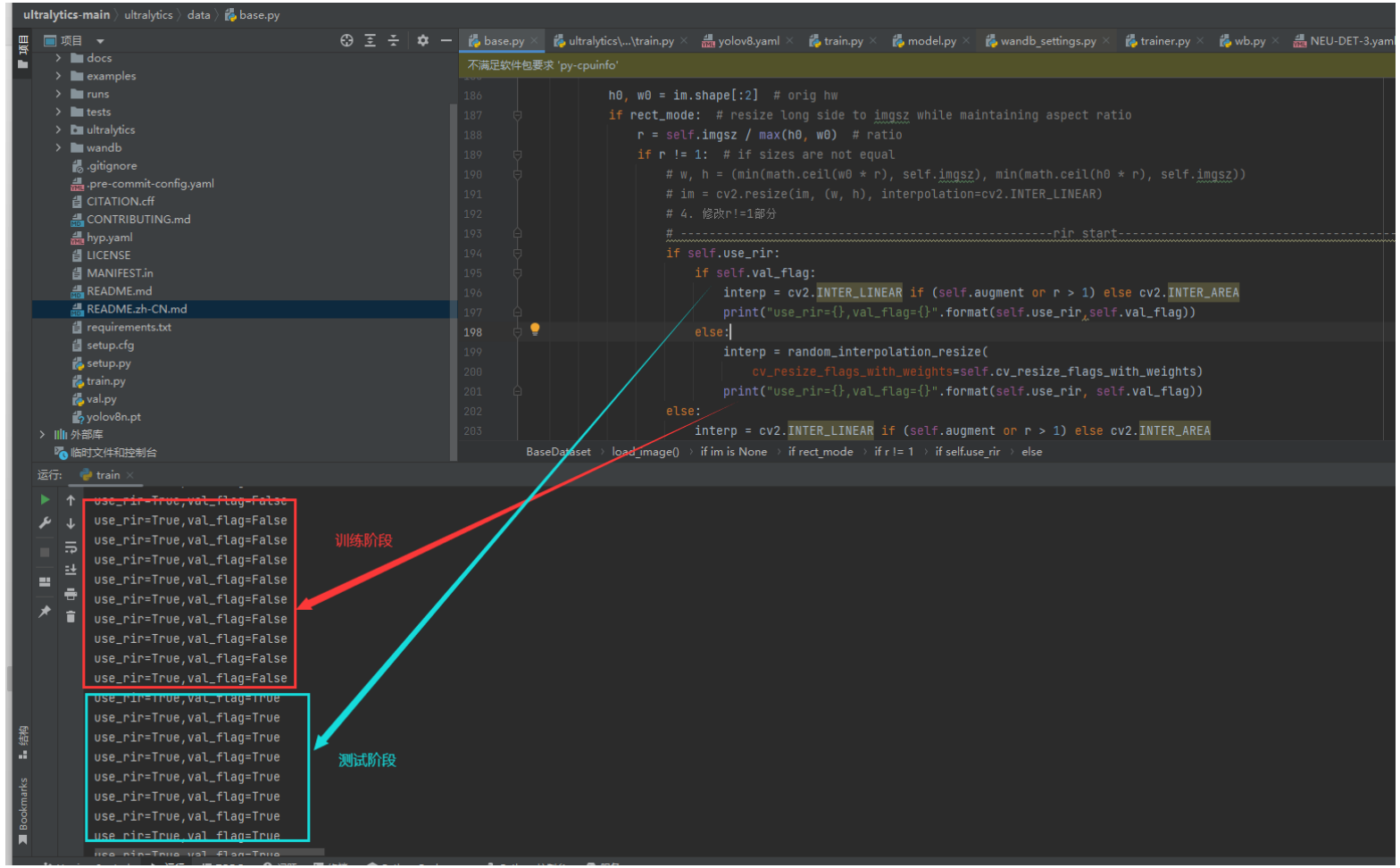




3.1 YOLOv8 添加步骤

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）

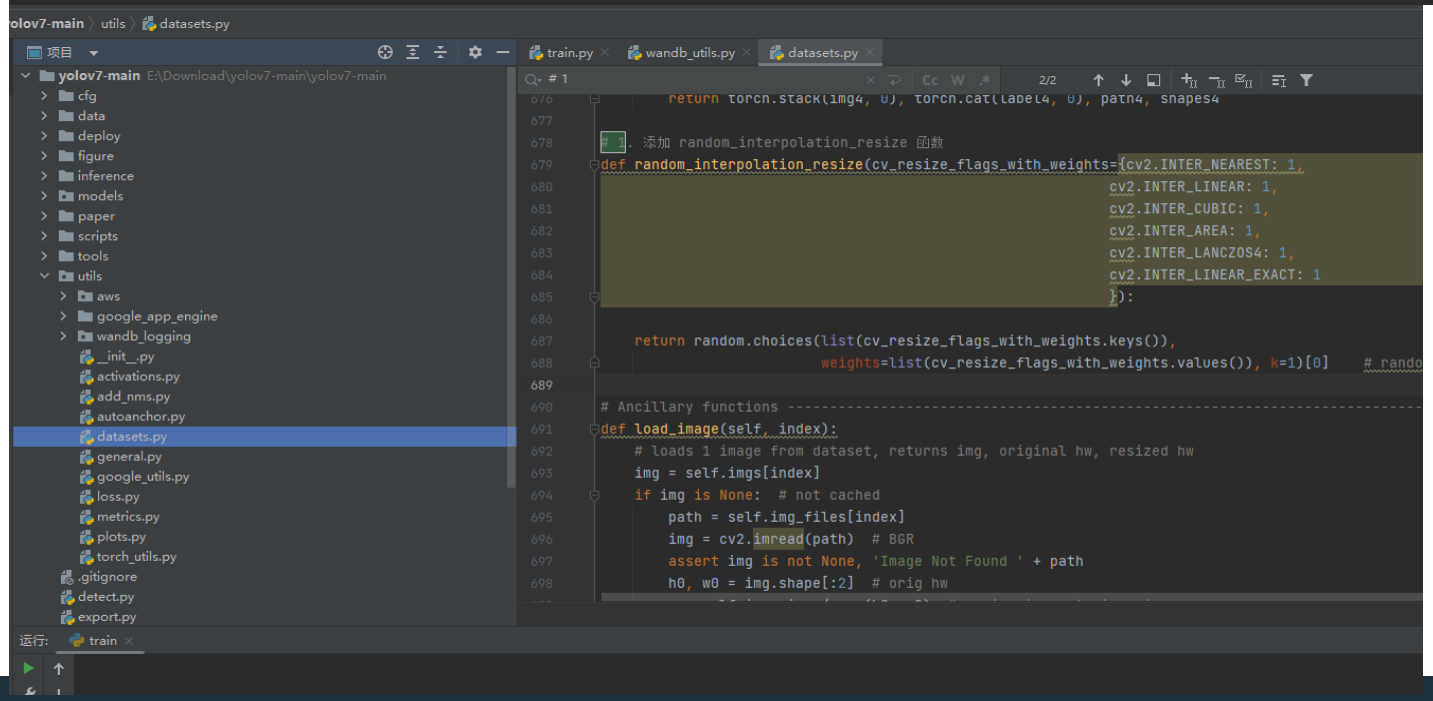
https://github.com/WongKinYiu/yolov7

3.3 YOLOv7 添加步骤

1.utils/dataset.py

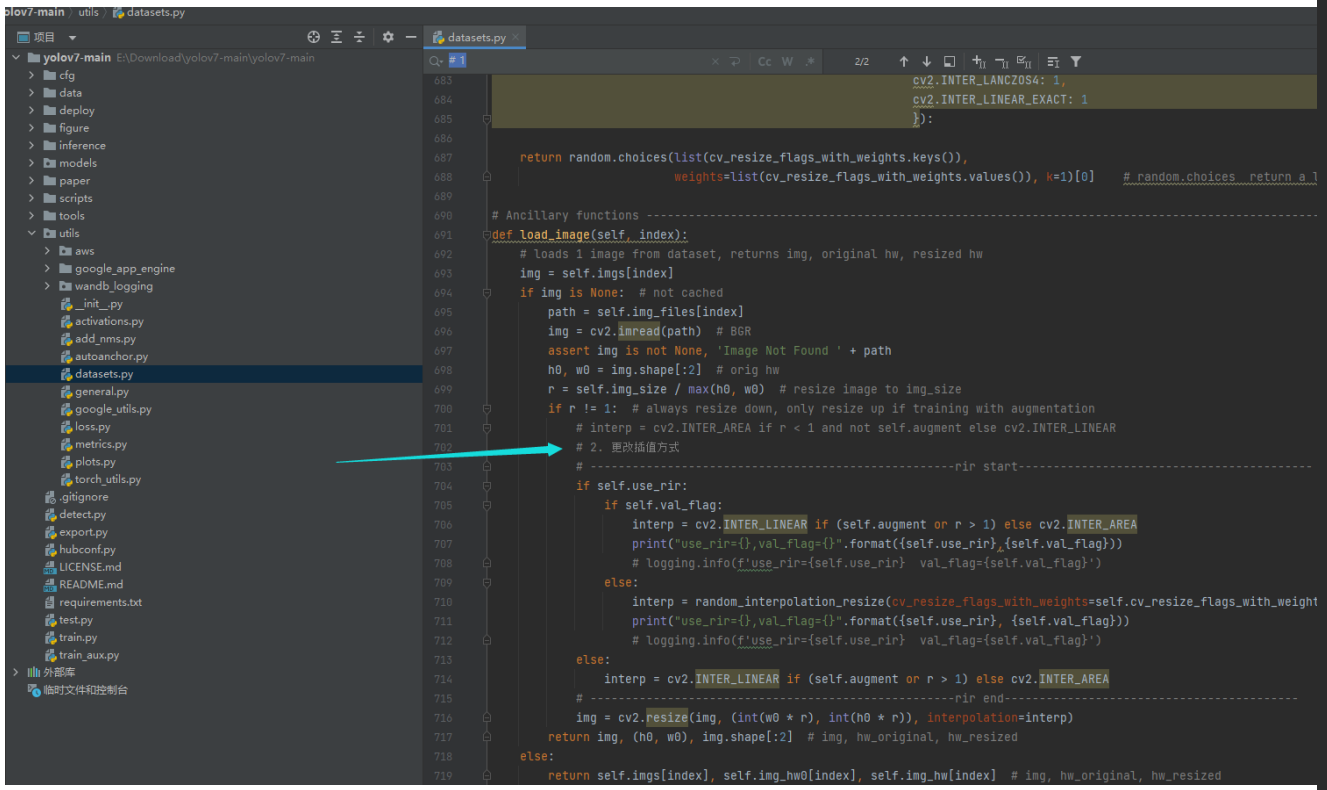
（1） 添加 random_interpolation_resize 函数

```
# 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

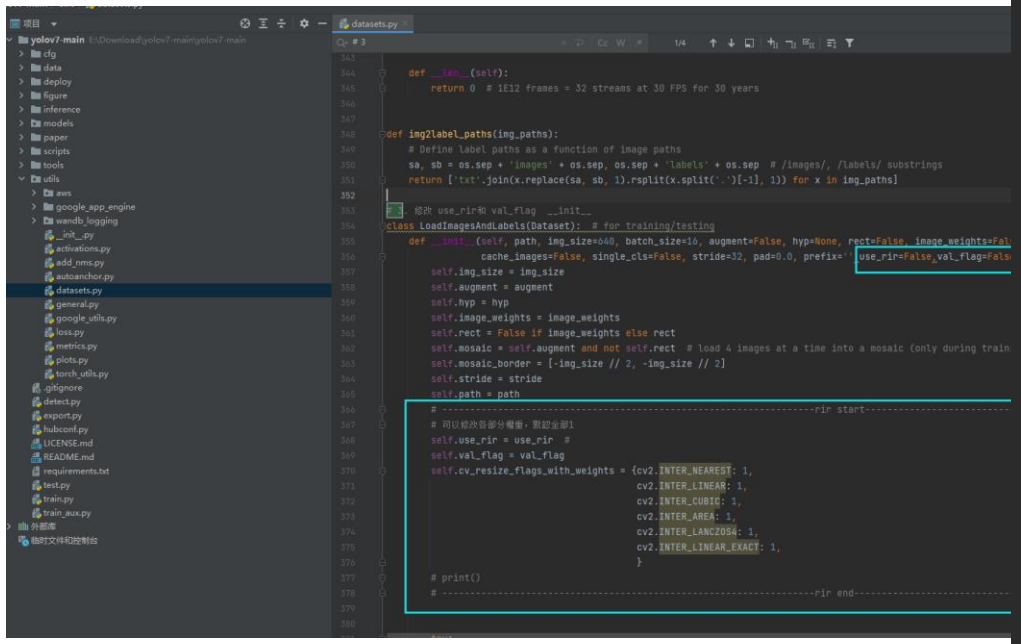



```
# Ancillary functions ----------
def load_image(self, index):
    # loads 1 image from dataset, returns img, original hw, resized hw
    img = self.imgs[index]
    if img is None:  # not cached
        path = self.img_files[index]
        img = cv2.imread(path)  # BGR
        assert img is not None, '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__

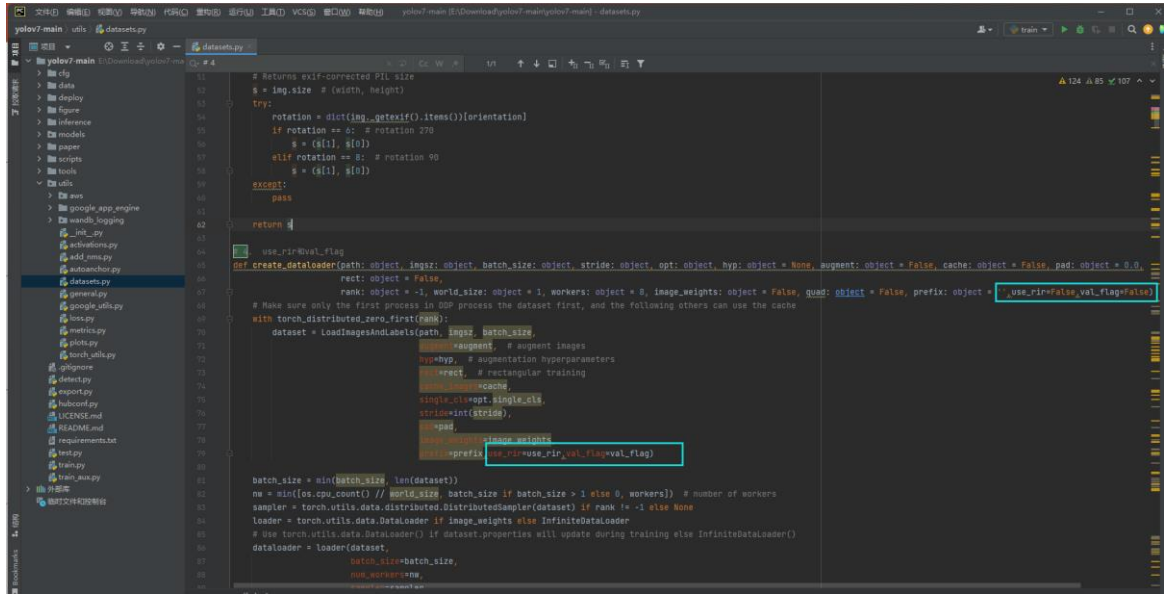


```
# 3. 修改 use_rir和 val_flag  __init__
class LoadImagesAndLabels(Dataset):  # for training/testing
    def __init__(self, path, img_size=640, batch_size=16, augment=False, hyp=None, rect=False, image_weights=False,
                 cache_images=False, single_cls=False, stride=32, pad=0.0, prefix='',use_rir=False,val_flag=False):
        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----------------------------
        # 可以修改各部分權重，默認全部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__

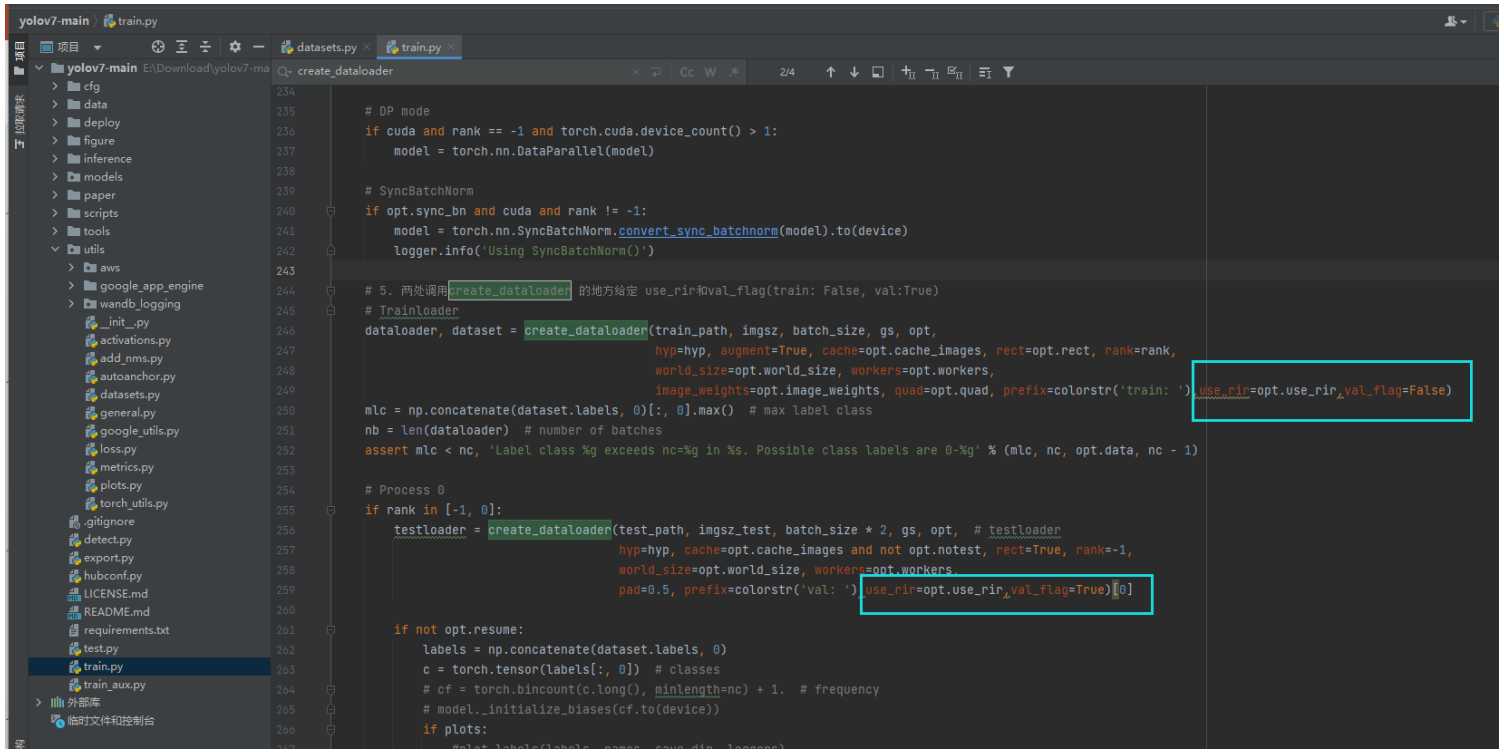


```
# 4.  use_rir和val_flag
def create_dataloader(path: object, imgsz: object, batch_size: object, stride: object, opt: object, hyp: object = None, aug
cache: object = False, pad: object = 0.0,
                      rect: object = False,
                      rank: object = -1, world_size: object = 1, workers: object = 8, image_weights: object = False, quad: object
",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, imgsz, 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




```
# 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)
```

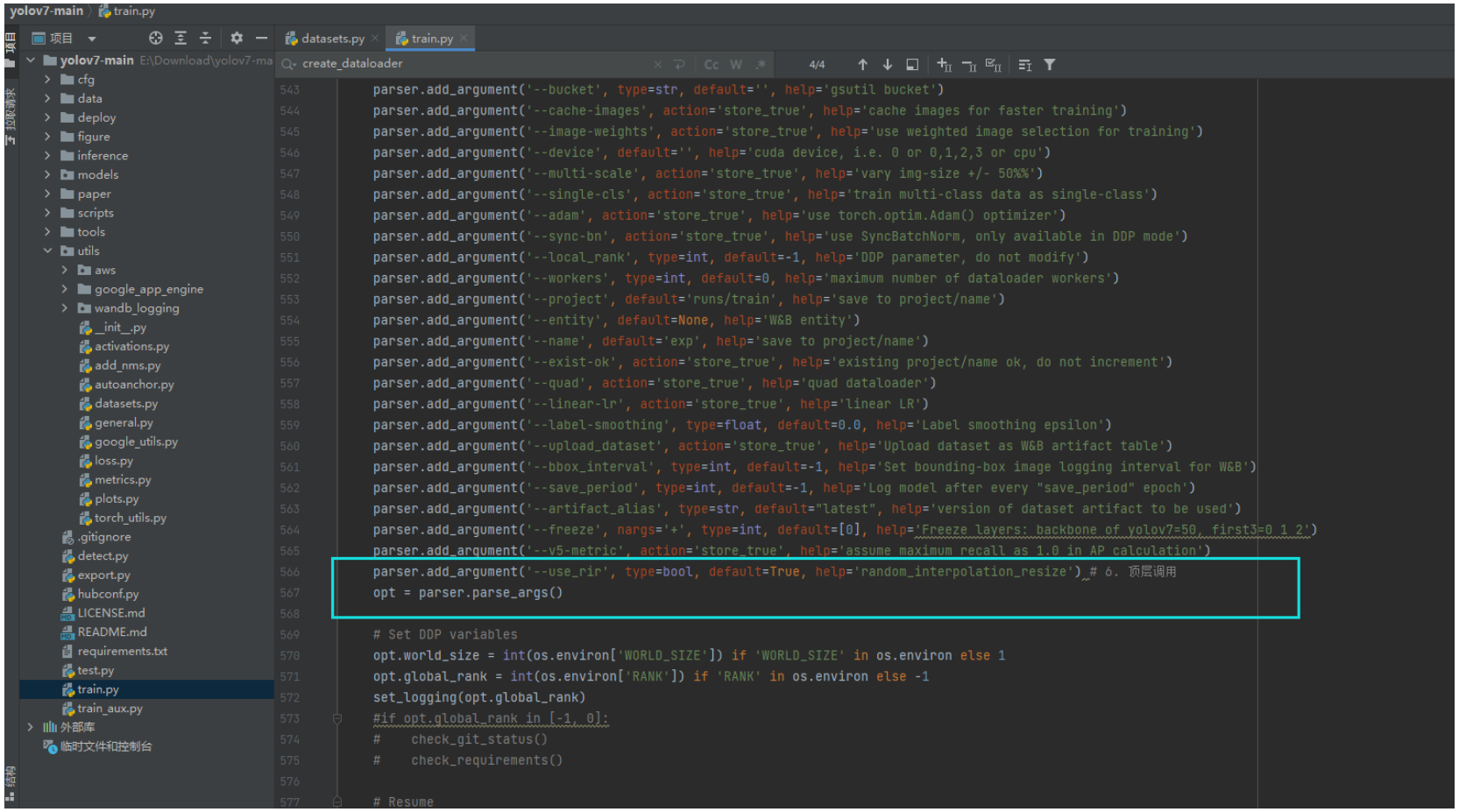
```
testloader = create_dataloader(test_path, imgsz_test, batch_size * 2, gs, opt,  # testloader
                               hyp=hyp, cache=opt.cache_images and not opt.notest, rect=True, rank=-1,
                               world_size=opt.world_size, workers=opt.workers,
                               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
```



3.3 YOLOv7 添加步骤

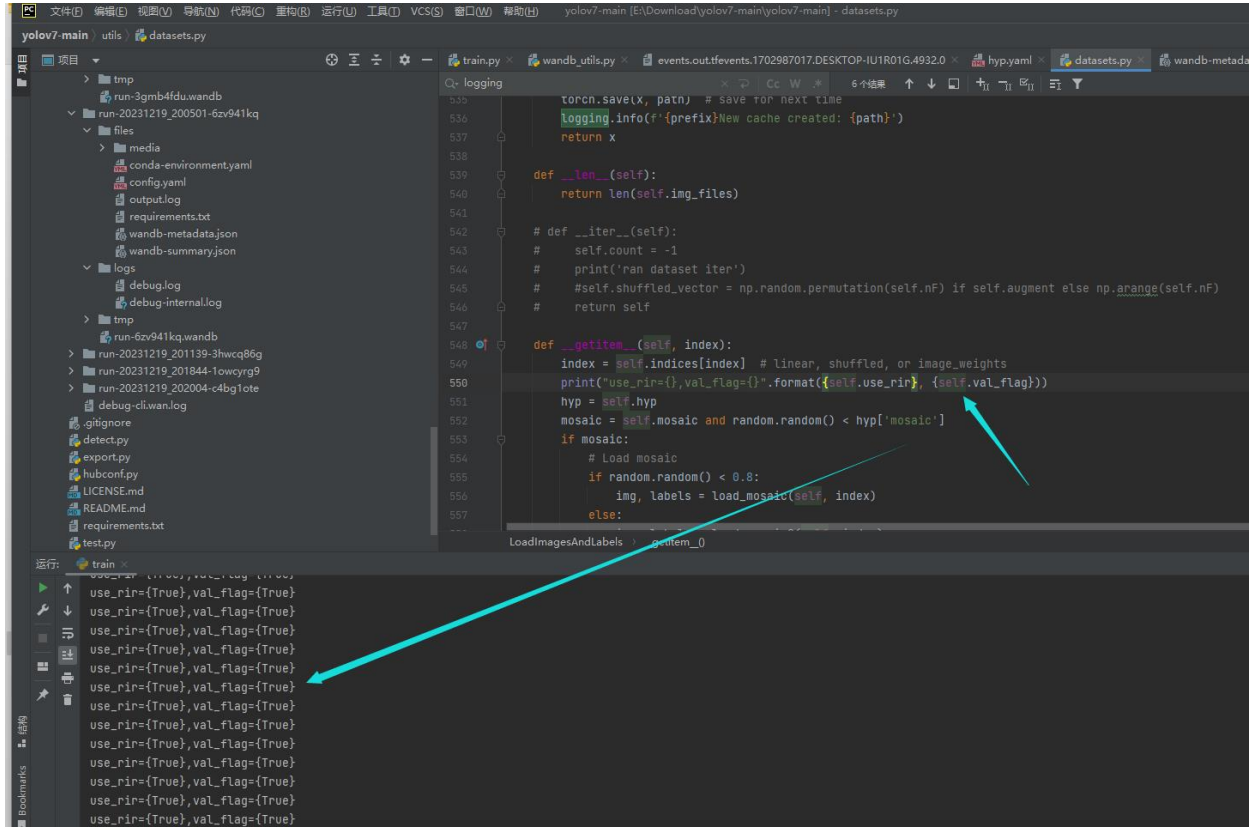
3.验证

（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__



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