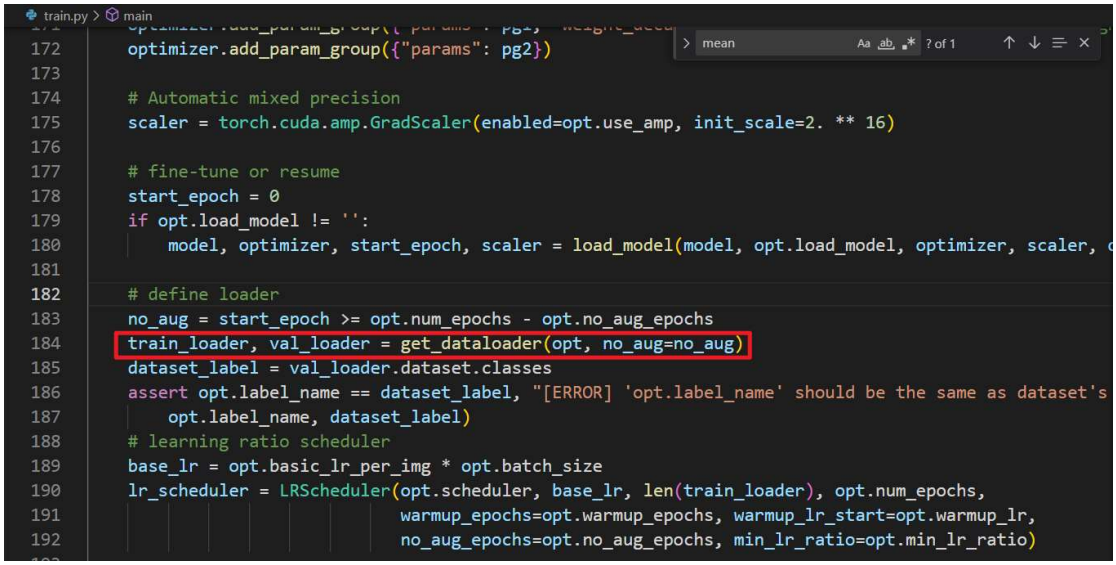
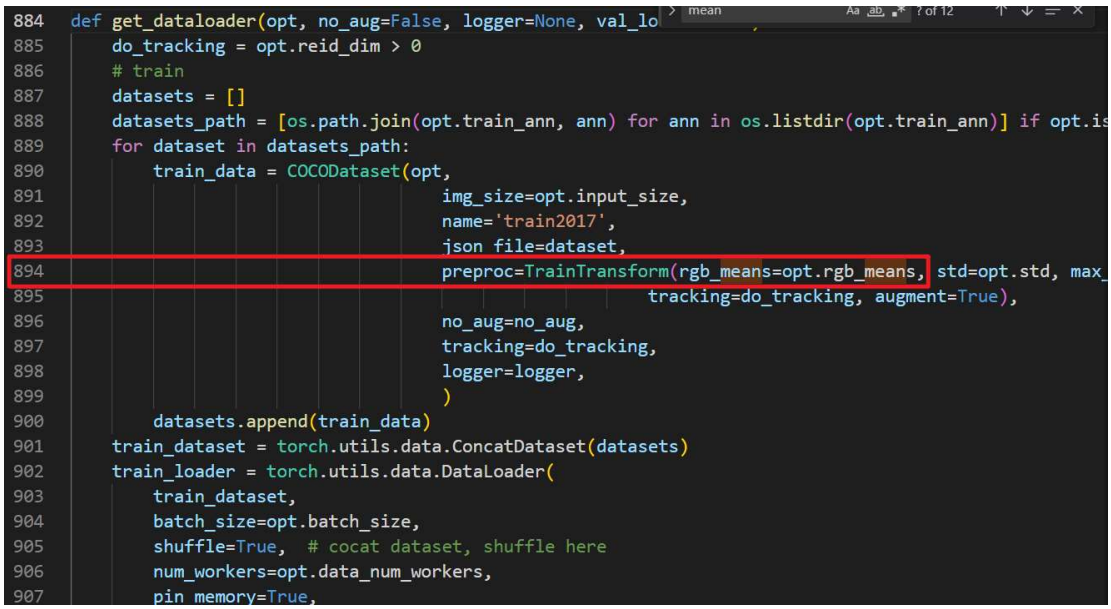
1.train 添加 Normalize 参考过程

1）train.py



```
train.py > main
172     optimizer.add_param_group({"params": pg2})
173
174     # Automatic mixed precision
175     scaler = torch.cuda.amp.GradScaler(enabled=opt.use_amp, init_scale=2. ** 16)
176
177     # fine-tune or resume
178     start_epoch = 0
179     if opt.load_model != '':
180         model, optimizer, start_epoch, scaler = load_model(model, opt.load_model, optimizer, scaler,
181
182     # define loader
183     no_aug = start_epoch >= opt.num_epochs - opt.no_aug_epochs
184     train_loader, val_loader = get_dataloader(opt, no_aug=no_aug)
185     dataset_label = val_loader.dataset.classes
186     assert opt.label_name == dataset_label, "[ERROR] 'opt.label_name' should be the same as dataset's
187         opt.label_name, dataset_label)
188     # learning ratio scheduler
189     base_lr = opt.basic_lr_per_img * opt.batch_size
190     lr_scheduler = LRScheduler(opt.scheduler, base_lr, len(train_loader), opt.num_epochs,
191                                warmup_epochs=opt.warmup_epochs, warmup_lr_start=opt.warmup_lr,
192                                no_aug_epochs=opt.no_aug_epochs, min_lr_ratio=opt.min_lr_ratio)
```

2）data\dataset.py



```
884 def get_dataloader(opt, no_aug=False, logger=None, val_lo
885     do_tracking = opt.reid_dim > 0
886     # train
887     datasets = []
888     datasets_path = [os.path.join(opt.train_ann, ann) for ann in os.listdir(opt.train_ann)] if opt.is
889     for dataset in datasets_path:
890         train_data = COCODataset(opt,
891                                  img_size=opt.input_size,
892                                  name='train2017',
893                                  json_file=dataset,
894                                  preproc=TrainTransform(rgb_means=opt.rgb_means, std=opt.std, max_
895                                                         tracking=do_tracking, augment=True),
896                                  no_aug=no_aug,
897                                  tracking=do_tracking,
898                                  logger=logger,
899                                  )
900         datasets.append(train_data)
901     train_dataset = torch.utils.data.ConcatDataset(datasets)
902     train_loader = torch.utils.data.DataLoader(
903         train_dataset,
904         batch_size=opt.batch_size,
905         shuffle=True,  # cocat dataset, shuffle here
906         num_workers=opt.data_num_workers,
907         pin_memory=True,
```

3）data\data_augment.py

```
190     padded_img = padded_img.transpose(swap)
191     padded_img = np.ascontiguousarray(padded_img, dtype=np.float32)
192     return padded_img, r
193
194
195 class TrainTransform:
196     def __init__(self, rgb_means=None, std=None, tracking=False, max_labels=50, augment=True):
197         self.means = rgb_means
198         self.std = std
199         self.tracking = tracking
200         self.max_labels = max_labels
201         self.augment = augment
202         self.color_augmentor = ColorDistort()
203
204     def __call__(self, image, targets, input_dim):
205         assert targets.shape[1] == 6 if self.tracking else 5
206         lshape = targets.shape[1]
207
208         boxes = targets[:, :4].copy()
209         labels = targets[:, 4].copy()
210         if self.tracking:
211             tracking_id = targets[:, 5].copy()
212
213         if len(boxes) == 0:
214             targets = np.zeros((self.max_labels, lshape), dtype=np.float32)
```

mean Aa ab .* ? of 14

data > data_augment.py > preproc

```
169
170 def preproc(image, input_size, mean, std, swap=(2, 0, 1)):
171     if len(image.shape) == 3:
172         padded_img = np.ones((input_size[0], input_size[1], 3)) * 114.0
173     else:
174         padded_img = np.ones(input_size) * 114.0
175     img = np.array(image)
176     r = min(input_size[0] / img.shape[0], input_size[1] / img.shape[1])
177     resized_img = cv2.resize(
178         img,
179         (int(img.shape[1] * r), int(img.shape[0] * r)),
180         interpolation=cv2.INTER_LINEAR,
181     ).astype(np.float32)
182     padded_img[: int(img.shape[0] * r), : int(img.shape[1] * r)] = resized_img
183
184     padded_img = padded_img[:, :, ::-1]
185     padded_img /= 255.0
186     if mean is not None:
187         padded_img -= mean
188     if std is not None:
189         padded_img /= std
190     padded_img = padded_img.transpose(swap)
191     padded_img = np.ascontiguousarray(padded_img, dtype=np.float32)
192     return padded_img, r
193
```

2.Predict 添加 Normalize 参考过程

1）predict.py

```
72      img_list = ["/algdata02/minqiang.xu/Project/yolox-pytorch/test_img/0056_51.jpg"]#get_img_path(img_
73      assert len(img_list) != 0, "cannot find img in {}".format(img_dir)
74
75      detector = Detector(opt)
76      for index, image_path in enumerate(tqdm.tqdm(img_list)):
77          # print("-----------------------------")
78          # print("{}/{}, {}".format(index, len(img_list), image_path))
79
80          assert os.path.isfile(image_path), "cannot find {}".format(image_path)
81          img = cv2.imread(image_path)
82          s1 = time.time()
83          results = detector.run(img, vis_thresh=opt.vis_thresh, show_time=False)
84          # print("[pre_process + inference + post_process] time cost: {}s".format(time.time() - s1))
85          # print(results)
86          img = vis_result(img, results)
87          save_p = output + "/" + image_path.split("/")[-2]
88          mkdir(save_p)
89          save_img = save_p + "/" + os.path.basename(image_path)
90          cv2.imwrite(save_img, img)
91          # print("save image to {}".format(save_img))
```

2）models\yolox.py

```
98      def run(self, images, vis_thresh, show_time=False):
99          batch_img = True
100         if np.ndim(images) == 3:
101             (variable) batch_img: Literal[False]
102             batch_img = False
103
104         with torch.no_grad():
105             if show_time:
106                 s1 = time.time()
107
108             img_ratios, img_shape = [], []
109             inp_imgs = np.zeros([len(images), 3, self.opt.test_size[0], self.opt.test_size[1]], dtype=
110             for b_i, image in enumerate(images):
111                 img_shape.append(image.shape[:2])
112                 img, r = preproc(image, self.opt.test_size, self.opt.rgb_means, self.opt.std)
113                 inp_imgs[b_i] = img
114                 img_ratios.append(r)
115
116             if show_time:
117                 s2 = time.time()
118                 print("[pre_process] time {}".format(s2 - s1))
119
120             inp_imgs = torch.from_numpy(inp_imgs).to(self.opt.device)
```

3）data\data_augment.py

```
170 def preproc(image, input_size, mean, std, swap=(2, 0, 1)):
171     if len(image.shape) == 3:
172         padded_img = np.ones((input_size[0], input_size[1], 3)) * 114.0
173     else:
174         padded_img = np.ones(input_size) * 114.0
175     img = np.array(image)
176     r = min(input_size[0] / img.shape[0], input_size[1] / img.shape[1])
177     resized_img = cv2.resize(
178         img,
179         (int(img.shape[1] * r), int(img.shape[0] * r)),
180         interpolation=cv2.INTER_LINEAR,
181     ).astype(np.float32)
182     padded_img[: int(img.shape[0] * r), : int(img.shape[1] * r)] = resized_img
183
184     padded_img = padded_img[:, :, ::-1]
185     padded_img /= 255.0
186     if mean is not None:
187         padded_img -= mean
188     if std is not None:
189         padded_img /= std
190     padded_img = padded_img.transpose(swap)
191     padded_img = np.ascontiguousarray(padded_img, dtype=np.float32)
192     return padded_img, r
193
```