目标检测代码实现

1. 新建项目

1）在AIStudio环境下新建项目，并添加数据集（在数据集中搜索螺丝螺母，并挂到项目下）

2）执行下列代码进行预处理

```
# 解压螺丝螺母数据，并将数据处理成需要的格式
!cd data/data6045/ && unzip -qo lslm.zip && unzip -qo lslm-test.zip
!cd data/data6045/ && mv lslm/*.txt .
!cd data/data6045/ && mv lslm-test/*.txt .
!cd data/data6045/ && sed -i 's/^/lslm\//' train.txt
!cd data/data6045/ && sed -i 's/^/lslm-test\//' eval.txt
!cd data/data6045/ && awk '{print $2}' label_list.txt > label_list
!echo "解压完成."
```

2. 项目代码

```
# -*- coding: UTF-8 -*-
"""
训练常基于dark-net的YOLOv3网络，目标检测
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import os

os.environ["FLAGS_fraction_of_gpu_memory_to_use"] = '0.82'

import uuid
import numpy as np
import time
import six
import math
import random
import paddle
import paddle.fluid as fluid
import logging
import xml.etree.ElementTree
import codecs
import json

from paddle.fluid.initializer import MSRA
```

```
from paddle.fluid.param_attr import ParamAttr
from paddle.fluid.regularizer import L2Decay
from PIL import Image, ImageEnhance, ImageDraw

logger = None  # 日志对象

train_params = {
    "data_dir": "data/data6045",  # 数据目录
    "train_list": "train.txt",  # 训练集文件
    "eval_list": "eval.txt",
    "class_dim": -1,
    "label_dict": {},  # 标签字典
    "num_dict": {},
    "image_count": -1,
    "continue_train": True,  # 是否加载前一次的训练参数，接着训练
    "pretrained": False,  # 是否预训练
    "pretrained_model_dir": "./pretrained-model",
    "save_model_dir": "./yolo-model",  # 模型保存目录
    "model_prefix": "yolo-v3",  # 模型前缀
    "freeze_dir": "freeze_model",
    "use_tiny": True,  # 是否使用 裁剪 tiny 模型
    "max_box_num": 20,  # 一幅图上最多有多少个目标
    "num_epochs": 80,  # 训练轮次
    "train_batch_size": 32,  # 对于完整yolov3，每一批的训练样本不能太多，内存会炸掉；如果使用tiny，可以适当大一些
    "use_gpu": True,  # 是否使用GPU
    "yolo_cfg": {  # YOLO模型参数
        "input_size": [3, 448, 448],  # 原版的边长大小为608，为了提高训练速度和预测速度，此处压缩为448
        "anchors": [7, 10, 12, 22, 24, 17, 22, 45, 46, 33, 43, 88, 85, 66, 115, 146, 275, 240],  # 锚点??
        "anchor_mask": [[6, 7, 8], [3, 4, 5], [0, 1, 2]]
    },
    "yolo_tiny_cfg": {  # YOLO tiny 模型参数
        "input_size": [3, 256, 256],
        "anchors": [6, 8, 13, 15, 22, 34, 48, 50, 81, 100, 205, 191],
        "anchor_mask": [[3, 4, 5], [0, 1, 2]]
    },
    "ignore_thresh": 0.7,
    "mean_rgb": [127.5, 127.5, 127.5],
    "mode": "train",
    "multi_data_reader_count": 4,
    "apply_distort": True,  # 是否做图像扭曲增强
    "nms_top_k": 300,
    "nms_pos_k": 300,
    "valid_thresh": 0.01,
    "nms_thresh": 0.45,  # 非最大值抑制阈值
```

```
    "image_distort_strategy": {  # 图像扭曲策略
        "expand_prob": 0.5,  # 扩展比率
        "expand_max_ratio": 4,
        "hue_prob": 0.5,  # 色调
        "hue_delta": 18,
        "contrast_prob": 0.5,  # 对比度
        "contrast_delta": 0.5,
        "saturation_prob": 0.5,  # 饱和度
        "saturation_delta": 0.5,
        "brightness_prob": 0.5,  # 亮度
        "brightness_delta": 0.125
    },
    "sgd_strategy": {  # 梯度下降配置
        "learning_rate": 0.002,
        "lr_epochs": [30, 50, 65],  # 学习率衰减分段（3个数字分为4段）
        "lr_decay": [1, 0.5, 0.25, 0.1]  # 每段采用的学习率，对应lr_epochs参数4段
    },
    "early_stop": {
        "sample_frequency": 50,
        "successive_limit": 3,
        "min_loss": 2.5,
        "min_curr_map": 0.84
    }
}


def init_train_parameters():
    """
    初始化训练参数，主要是初始化图片数量，类别数
    :return:
    """
    file_list = os.path.join(train_params['data_dir'], train_params['train_list'])  # 训练集
    label_list = os.path.join(train_params['data_dir'], "label_list")  # 标签文件
    index = 0

    # codecs是专门用作编码转换通用模块
    with codecs.open(label_list, encoding='utf-8') as flist:
        lines = [line.strip() for line in flist]
        for line in lines:
            train_params['num_dict'][index] = line.strip()
            train_params['label_dict'][line.strip()] = index
            index += 1
        train_params['class_dim'] = index

```

```
    with codecs.open(file_list, encoding='utf-8') as flist:
        lines = [line.strip() for line in flist]
        train_params['image_count'] = len(lines)  # 图片数量


# 日志相关配置
def init_log_config():  # 初始化日志相关配置
    global logger

    logger = logging.getLogger()  # 创建日志对象
    logger.setLevel(logging.INFO)  # 设置日志级别
    log_path = os.path.join(os.getcwd(), 'logs')

    if not os.path.exists(log_path):  # 创建日志路径
        os.makedirs(log_path)

    log_name = os.path.join(log_path, 'train.log')  # 训练日志文件
    fh = logging.FileHandler(log_name, mode='w')  # 打开文件句柄
    fh.setLevel(logging.DEBUG)  # 设置级别

    formatter = logging.Formatter("%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s")
    fh.setFormatter(formatter)
    logger.addHandler(fh)


init_log_config()


# 定义YOLO3网络结构：darknet-53
class YOLOv3(object):
    def __init__(self, class_num, anchors, anchor_mask):
        self.outputs = []  # 网络最终模型
        self.downsample_ratio = 1  # 下采样率
        self.anchor_mask = anchor_mask  # 计算卷积核？？？
        self.anchors = anchors  # 锚点
        self.class_num = class_num  # 类别数量

        self.yolo_anchors = []
        self.yolo_classes = []

        for mask_pair in self.anchor_mask:
            mask_anchors = []
            for mask in mask_pair:
                mask_anchors.append(self.anchors[2 * mask])
                mask_anchors.append(self.anchors[2 * mask + 1])
            self.yolo_anchors.append(mask_anchors)
```

```
            self.yolo_classes.append(class_num)

    def name(self):
        return 'YOLOv3'

    # 获取anchors
    def get_anchors(self):
        return self.anchors

    # 获取anchor_mask
    def get_anchor_mask(self):
        return self.anchor_mask

    def get_class_num(self):
        return self.class_num

    def get_downsample_ratio(self):
        return self.downsample_ratio

    def get_yolo_anchors(self):
        return self.yolo_anchors

    def get_yolo_classes(self):
        return self.yolo_classes

    # 卷积正则化函数
    def conv_bn(self,
                input,  # 输入
                num_filters,  # 卷积核数量
                filter_size,  # 卷积核大小
                stride,  # 步幅
                padding,  # 填充
                use_cudnn=True):
        # 2d卷积操作
        conv = fluid.layers.conv2d(input=input,
                                   num_filters=num_filters,
                                   filter_size=filter_size,
                                   stride=stride,
                                   padding=padding,
                                   act=None,
                                   use_cudnn=use_cudnn,  # 是否使用
cudnn，cudnn利用cuda进行了加速处理

param_attr=ParamAttr(initializer=fluid.initializer.Normal(0., 0.02)),
                                   bias_attr=False)

```

```
        # batch_norm中的参数不需要参与正则化，所以主动使用正则系数为0的正则项屏蔽掉
        # 在batch_norm中使用leaky的话，只能使用默认的alpha=0.02；如果需要设值，必须提出去单独来
        # 正则化的目的，是为了防止过拟合，较小的L2值能防止过拟合
        param_attr = ParamAttr(initializer=fluid.initializer.Normal(0., 0.02),
                               regularizer=L2Decay(0.))
        bias_attr = ParamAttr(initializer=fluid.initializer.Constant(0.0),
                              regularizer=L2Decay(0.))
        out = fluid.layers.batch_norm(input=conv, act=None,
                                      param_attr=param_attr,
                                      bias_attr=bias_attr)
        # leaky_relu: Leaky ReLU是给所有负值赋予一个非零斜率
        out = fluid.layers.leaky_relu(out, 0.1)
        return out

    # 通过卷积实现降采样
    # 如：原始图片大小448*448，降采样后大小为 ((448+2)-3)/2 + 1 = 224
    def down_sample(self, input, num_filters, filter_size=3, stride=2, padding=1):
        self.downsample_ratio *= 2  # 降采样率
        return self.conv_bn(input,
                            num_filters=num_filters,
                            filter_size=filter_size,
                            stride=stride,
                            padding=padding)

    # 基本块：包含两个卷积/正则化层，一个残差块
    def basic_block(self, input, num_filters):
        conv1 = self.conv_bn(input, num_filters, filter_size=1, stride=1, padding=0)
        conv2 = self.conv_bn(conv1, num_filters * 2, filter_size=3, stride=1, padding=1)
        out = fluid.layers.elementwise_add(x=input, y=conv2, act=None)  # 计算H(x)=F(x)+x
        return out

    # 创建多个basic_block
    def layer_warp(self, input, num_filters, count):
        res_out = self.basic_block(input, num_filters)
        for j in range(1, count):
            res_out = self.basic_block(res_out, num_filters)
        return res_out

    # 上采样
```

```
243     def up_sample(self, input, scale=2):
244         # get dynamic upsample output shape
245         shape_nchw = fluid.layers.shape(input)  # 获取input的形状
246         shape_hw = fluid.layers.slice(shape_nchw, axes=[0], starts=
    [2], ends=[4])
247         shape_hw.stop_gradient = True
248         in_shape = fluid.layers.cast(shape_hw, dtype='int32')
249         out_shape = in_shape * scale  # 计算输出数据形状
250         out_shape.stop_gradient = True
251 
252         # reisze by actual_shape
253         # 矩阵放大
254         out = fluid.layers.resize_nearest(input=input,
255                                           scale=scale,
256                                           actual_shape=out_shape)
257         return out
258 
259     def yolo_detection_block(self, input, num_filters):
260         assert num_filters % 2 == 0, "num_filters {} cannot be
    divided by 2".format(num_filters)
261 
262         conv = input
263         for j in range(2):
264             conv = self.conv_bn(conv, num_filters, filter_size=1,
    stride=1, padding=0)
265             conv = self.conv_bn(conv, num_filters * 2, filter_size=3,
    stride=1, padding=1)
266         route = self.conv_bn(conv, num_filters, filter_size=1,
    stride=1, padding=0)
267         tip = self.conv_bn(route, num_filters * 2, filter_size=3,
    stride=1, padding=1)
268         return route, tip
269 
270     # 搭建网络模型 darknet-53
271     def net(self, img):
272         stages = [1, 2, 8, 8, 4]
273         assert len(self.anchor_mask) <= len(stages), "anchor masks
    can't bigger than down_sample times"
274         # 第一个卷积层: 256*256
275         conv1 = self.conv_bn(img, num_filters=32, filter_size=3,
    stride=1, padding=1)
276         # 第二个卷积层: 128*128
277         downsample_ = self.down_sample(conv1, conv1.shape[1] * 2)  #
    第二个参数为卷积核数量
278         blocks = []
279 
280         # 循环创建basic_block组
```

```
        for i, stage_count in enumerate(stages):
            block = self.layer_warp(downsample_,  # 输入数据
                                    32 * (2 ** i),  # 卷积核数量
                                    stage_count)  # 基本块数量
            blocks.append(block)
            if i < len(stages) - 1:  # 如果不是最后一组，做降采样
                downsample_ = self.down_sample(block, block.shape[1] * 2)
        blocks = blocks[-1:-4:-1]  # 取倒数三层，并且逆序，后面跨层级联需要

        # yolo detector
        for i, block in enumerate(blocks):
            # yolo中跨视域链接
            if i > 0:
                block = fluid.layers.concat(input=[route, block], axis=1)  # 连接route和block，按行

            route, tip = self.yolo_detection_block(block,  # 输入
                                                   num_filters=512 // (2 ** i))  # 卷积核数量

            param_attr = ParamAttr(initializer=fluid.initializer.Normal(0., 0.02))
            bias_attr = ParamAttr(initializer=fluid.initializer.Constant(0.0), regularizer=L2Decay(0.))
            block_out = fluid.layers.conv2d(input=tip,
                                            # 5 elements represent x|y|h|w|score

 num_filters=len(self.anchor_mask[i]) * (self.class_num + 5),
                                            filter_size=1,
                                            stride=1,
                                            padding=0,
                                            act=None,
                                            param_attr=param_attr,
                                            bias_attr=bias_attr)
            self.outputs.append(block_out)

            # 为了跨视域链接，差值方式提升特征图尺寸
            if i < len(blocks) - 1:
                route = self.conv_bn(route, 256 // (2 ** i), filter_size=1, stride=1, padding=0)
                route = self.up_sample(route)  # 上采样

        return self.outputs

```

```
319
320 class YOLOv3Tiny(object):
321     def __init__(self, class_num, anchors, anchor_mask):
322         self.outputs = []
323         self.downsample_ratio = 1
324         self.anchor_mask = anchor_mask
325         self.anchors = anchors
326         self.class_num = class_num
327
328         self.yolo_anchors = []
329         self.yolo_classes = []
330         for mask_pair in self.anchor_mask:
331             mask_anchors = []
332             for mask in mask_pair:
333                 mask_anchors.append(self.anchors[2 * mask])
334                 mask_anchors.append(self.anchors[2 * mask + 1])
335             self.yolo_anchors.append(mask_anchors)
336             self.yolo_classes.append(class_num)
337
338     def name(self):
339         return 'YOLOv3-tiny'
340
341     def get_anchors(self):
342         return self.anchors
343
344     def get_anchor_mask(self):
345         return self.anchor_mask
346
347     def get_class_num(self):
348         return self.class_num
349
350     def get_downsample_ratio(self):
351         return self.downsample_ratio
352
353     def get_yolo_anchors(self):
354         return self.yolo_anchors
355
356     def get_yolo_classes(self):
357         return self.yolo_classes
358
359     def conv_bn(self,
360                 input,
361                 num_filters,
362                 filter_size,
363                 stride,
364                 padding,
365                 num_groups=1,
```

```
                use_cudnn=True):
        conv = fluid.layers.conv2d(
            input=input,
            num_filters=num_filters,
            filter_size=filter_size,
            stride=stride,
            padding=padding,
            act=None,
            groups=num_groups,
            use_cudnn=use_cudnn,

 param_attr=ParamAttr(initializer=fluid.initializer.Normal(0.,
0.02)),
            bias_attr=False)

        # batch_norm中的参数不需要参与正则化，所以主动使用正则系数为0的正则项屏
蔽掉
        out = fluid.layers.batch_norm(
            input=conv, act='relu',

 param_attr=ParamAttr(initializer=fluid.initializer.Normal(0., 0.02),
regularizer=L2Decay(0.)),

 bias_attr=ParamAttr(initializer=fluid.initializer.Constant(0.0),
regularizer=L2Decay(0.)))

        return out

    def depthwise_conv_bn(self, input, filter_size=3, stride=1,
padding=1):
        num_filters = input.shape[1]
        return self.conv_bn(input,
                            num_filters=num_filters,
                            filter_size=filter_size,
                            stride=stride,
                            padding=padding,
                            num_groups=num_filters)

    def down_sample(self, input, pool_size=2, pool_stride=2):
        self.downsample_ratio *= 2
        return fluid.layers.pool2d(input=input, pool_type='max',
pool_size=pool_size,
                                   pool_stride=pool_stride)

    def basic_block(self, input, num_filters):
        conv1 = self.conv_bn(input, num_filters, filter_size=3,
stride=1, padding=1)
```

```
403         out = self.down_sample(conv1)
404         return out
405 
406     def up_sample(self, input, scale=2):
407         # get dynamic upsample output shape
408         shape_nchw = fluid.layers.shape(input)
409         shape_hw = fluid.layers.slice(shape_nchw, axes=[0], starts=
    [2], ends=[4])
410         shape_hw.stop_gradient = True
411         in_shape = fluid.layers.cast(shape_hw, dtype='int32')
412         out_shape = in_shape * scale
413         out_shape.stop_gradient = True
414 
415         # reisze by actual_shape
416         out = fluid.layers.resize_nearest(
417             input=input,
418             scale=scale,
419             actual_shape=out_shape)
420         return out
421 
422     def yolo_detection_block(self, input, num_filters):
423         route = self.conv_bn(input, num_filters, filter_size=1,
    stride=1, padding=0)
424         tip = self.conv_bn(route, num_filters * 2, filter_size=3,
    stride=1, padding=1)
425         return route, tip
426 
427     def net(self, img):
428         # darknet-tiny
429         stages = [16, 32, 64, 128, 256, 512]
430         assert len(self.anchor_mask) <= len(stages), "anchor masks
    can't bigger than down_sample times"
431         # 256x256
432         tmp = img
433         blocks = []
434         for i, stage_count in enumerate(stages):
435             if i == len(stages) - 1:
436                 block = self.conv_bn(tmp, stage_count, filter_size=3,
    stride=1, padding=1)
437                 blocks.append(block)
438                 block = self.depthwise_conv_bn(blocks[-1])
439                 block = self.depthwise_conv_bn(blocks[-1])
440                 block = self.conv_bn(blocks[-1], stage_count * 2,
    filter_size=1, stride=1, padding=0)
441                 blocks.append(block)
442             else:
443                 tmp = self.basic_block(tmp, stage_count)
```

```
                blocks.append(tmp)

        blocks = [blocks[-1], blocks[3]]

        # yolo detector
        for i, block in enumerate(blocks):
            # yolo 中跨视域链接
            if i > 0:
                block = fluid.layers.concat(input=[route, block],
axis=1)
            if i < 1:
                route, tip = self.yolo_detection_block(block,
num_filters=256 // (2 ** i))
            else:
                tip = self.conv_bn(block, num_filters=256,
filter_size=3, stride=1, padding=1)

            param_attr =
ParamAttr(initializer=fluid.initializer.Normal(0., 0.02))
            bias_attr =
ParamAttr(initializer=fluid.initializer.Constant(0.0),
regularizer=L2Decay(0.))
            block_out = fluid.layers.conv2d(input=tip,
                                            # 5 elements represent
x|y|h|w|score

 num_filters=len(self.anchor_mask[i]) * (self.class_num + 5),
                                            filter_size=1,
                                            stride=1,
                                            padding=0,
                                            act=None,
                                            param_attr=param_attr,
                                            bias_attr=bias_attr)
            self.outputs.append(block_out)
            # 为了跨视域链接，差值方式提升特征图尺寸
            if i < len(blocks) - 1:
                route = self.conv_bn(route, 128 // (2 ** i),
filter_size=1, stride=1, padding=0)
                route = self.up_sample(route)

        return self.outputs


def get_yolo(is_tiny, class_num, anchors, anchor_mask):
    if is_tiny:
        return YOLOv3Tiny(class_num, anchors, anchor_mask)
    else:
```

```
482         return YOLOv3(class_num, anchors, anchor_mask)
483
484
485 class Sampler(object):
486     """
487     采样器，用于扣取采样
488     """
489
490     def __init__(self, max_sample, max_trial, min_scale, max_scale,
491                  min_aspect_ratio, max_aspect_ratio,
min_jaccard_overlap,
492                  max_jaccard_overlap):
493         self.max_sample = max_sample
494         self.max_trial = max_trial
495         self.min_scale = min_scale
496         self.max_scale = max_scale
497         self.min_aspect_ratio = min_aspect_ratio
498         self.max_aspect_ratio = max_aspect_ratio
499         self.min_jaccard_overlap = min_jaccard_overlap
500         self.max_jaccard_overlap = max_jaccard_overlap
501
502
503 class bbox(object):
504     """
505     外界矩形框
506     """
507
508     def __init__(self, xmin, ymin, xmax, ymax):
509         self.xmin = xmin
510         self.ymin = ymin
511         self.xmax = xmax
512         self.ymax = ymax
513
514
515 # 坐标转换，由[x1, y1, w, h]转换为[center_x, center_y, w, h]
516 # 并转换为范围在[0, 1]之间的相对坐标
517 def box_to_center_relative(box, img_height, img_width):
518     """
519     Convert COCO annotations box with format [x1, y1, w, h] to
520     center mode [center_x, center_y, w, h] and divide image width
521     and height to get relative value in range[0, 1]
522     """
523     assert len(box) == 4, "box should be a len(4) list or tuple"
524     x, y, w, h = box
525
526     x1 = max(x, 0)
527     x2 = min(x + w - 1, img_width - 1)
```

```
    y1 = max(y, 0)
    y2 = min(y + h - 1, img_height - 1)

    x = (x1 + x2) / 2 / img_width  # x中心坐标
    y = (y1 + y2) / 2 / img_height  # y中心坐标
    w = (x2 - x1) / img_width  # 框宽度/图片总宽度
    h = (y2 - y1) / img_height  # 框高度/图片总高度

    return np.array([x, y, w, h])


# 调整图像大小
def resize_img(img, sampled_labels, input_size):
    target_size = input_size
    img = img.resize((target_size[1], target_size[2]),
Image.BILINEAR)
    return img


# 计算交并比
def box_iou_xywh(box1, box2):
    assert box1.shape[-1] == 4, "Box1 shape[-1] should be 4."
    assert box2.shape[-1] == 4, "Box2 shape[-1] should be 4."

    # 取两个框的坐标
    b1_x1, b1_x2 = box1[:, 0] - box1[:, 2] / 2, box1[:, 0] + box1[:,
2] / 2
    b1_y1, b1_y2 = box1[:, 1] - box1[:, 3] / 2, box1[:, 1] + box1[:,
3] / 2
    b2_x1, b2_x2 = box2[:, 0] - box2[:, 2] / 2, box2[:, 0] + box2[:,
2] / 2
    b2_y1, b2_y2 = box2[:, 1] - box2[:, 3] / 2, box2[:, 1] + box2[:,
3] / 2

    inter_x1 = np.maximum(b1_x1, b2_x1)
    inter_x2 = np.minimum(b1_x2, b2_x2)
    inter_y1 = np.maximum(b1_y1, b2_y1)
    inter_y2 = np.minimum(b1_y2, b2_y2)
    inter_w = inter_x2 - inter_x1 + 1  # 相交部分宽度
    inter_h = inter_y2 - inter_y1 + 1  # 相交部分高度
    inter_w[inter_w < 0] = 0
    inter_h[inter_h < 0] = 0

    inter_area = inter_w * inter_h  # 相交面积
    b1_area = (b1_x2 - b1_x1 + 1) * (b1_y2 - b1_y1 + 1)  # 框1的面积
    b2_area = (b2_x2 - b2_x1 + 1) * (b2_y2 - b2_y1 + 1)  # 框2的面积

```

```
    return inter_area / (b1_area + b2_area - inter_area)  # 相集面积/并集面积


# box裁剪
def box_crop(boxes, labels, crop, img_shape):
    x, y, w, h = map(float, crop)
    im_w, im_h = map(float, img_shape)

    boxes = boxes.copy()
    boxes[:, 0], boxes[:, 2] = (boxes[:, 0] - boxes[:, 2] / 2) * im_w, (boxes[:, 0] + boxes[:, 2] / 2) * im_w
    boxes[:, 1], boxes[:, 3] = (boxes[:, 1] - boxes[:, 3] / 2) * im_h, (boxes[:, 1] + boxes[:, 3] / 2) * im_h

    crop_box = np.array([x, y, x + w, y + h])
    centers = (boxes[:, :2] + boxes[:, 2:]) / 2.0
    mask = np.logical_and(crop_box[:2] <= centers, centers <= crop_box[2:]).all(axis=1)

    boxes[:, :2] = np.maximum(boxes[:, :2], crop_box[:2])
    boxes[:, 2:] = np.minimum(boxes[:, 2:], crop_box[2:])
    boxes[:, :2] -= crop_box[:2]
    boxes[:, 2:] -= crop_box[:2]

    mask = np.logical_and(mask, (boxes[:, :2] < boxes[:, 2:]).all(axis=1))
    boxes = boxes * np.expand_dims(mask.astype('float32'), axis=1)
    labels = labels * mask.astype('float32')
    boxes[:, 0], boxes[:, 2] = (boxes[:, 0] + boxes[:, 2]) / 2 / w, (boxes[:, 2] - boxes[:, 0]) / w
    boxes[:, 1], boxes[:, 3] = (boxes[:, 1] + boxes[:, 3]) / 2 / h, (boxes[:, 3] - boxes[:, 1]) / h

    return boxes, labels, mask.sum()


# 图像增加：对比度，饱和度，明暗，颜色，扩张
def random_brightness(img):  # 亮度
    prob = np.random.uniform(0, 1)

    if prob < train_params['image_distort_strategy']['brightness_prob']:
        brightness_delta = train_params['image_distort_strategy']['brightness_delta']  # 默认值0.125
        delta = np.random.uniform(-brightness_delta, brightness_delta) + 1  # 产生均匀分布随机值
```

```
        img = ImageEnhance.Brightness(img).enhance(delta)  # 调整图像亮度

    return img


def random_contrast(img):  # 对比度
    prob = np.random.uniform(0, 1)

    if prob < train_params['image_distort_strategy']['contrast_prob']:
        contrast_delta = train_params['image_distort_strategy']['contrast_delta']
        delta = np.random.uniform(-contrast_delta, contrast_delta) + 1
        img = ImageEnhance.Contrast(img).enhance(delta)

    return img


def random_saturation(img):  # 饱和度
    prob = np.random.uniform(0, 1)

    if prob < train_params['image_distort_strategy']['saturation_prob']:
        saturation_delta = train_params['image_distort_strategy']['saturation_delta']
        delta = np.random.uniform(-saturation_delta, saturation_delta) + 1
        img = ImageEnhance.Color(img).enhance(delta)

    return img


def random_hue(img):  # 色调
    prob = np.random.uniform(0, 1)

    if prob < train_params['image_distort_strategy']['hue_prob']:
        hue_delta = train_params['image_distort_strategy']['hue_delta']
        delta = np.random.uniform(-hue_delta, hue_delta)
        img_hsv = np.array(img.convert('HSV'))
        img_hsv[:, :, 0] = img_hsv[:, :, 0] + delta
        img = Image.fromarray(img_hsv, mode='HSV').convert('RGB')

    return img

```

```
646
647 def distort_image(img):  # 图像扭曲
648     prob = np.random.uniform(0, 1)
649     # Apply different distort order
650     if prob > 0.5:
651         img = random_brightness(img)
652         img = random_contrast(img)
653         img = random_saturation(img)
654         img = random_hue(img)
655     else:
656         img = random_brightness(img)
657         img = random_saturation(img)
658         img = random_hue(img)
659         img = random_contrast(img)
660     return img
661
662
663 # 随机裁剪
664 def random_crop(img, boxes, labels, scales=[0.3, 1.0], max_ratio=2.0,
    constraints=None, max_trial=50):
665     if random.random() > 0.6:
666         return img, boxes, labels
667     if len(boxes) == 0:
668         return img, boxes, labels
669
670     if not constraints:
671         constraints = [(0.1, 1.0),
672                        (0.3, 1.0),
673                        (0.5, 1.0),
674                        (0.7, 1.0),
675                        (0.9, 1.0),
676                        (0.0, 1.0)]  # 最小/最大交并比值
677
678     w, h = img.size
679     crops = [(0, 0, w, h)]
680
681     for min_iou, max_iou in constraints:
682         for _ in range(max_trial):
683             scale = random.uniform(scales[0], scales[1])
684             aspect_ratio = random.uniform(max(1 / max_ratio, scale *
    scale), \
685                                           min(max_ratio, 1 / scale /
    scale))
686             crop_h = int(h * scale / np.sqrt(aspect_ratio))
687             crop_w = int(w * scale * np.sqrt(aspect_ratio))
688             crop_x = random.randrange(w - crop_w)
689             crop_y = random.randrange(h - crop_h)
```

```
            crop_box = np.array([[
                (crop_x + crop_w / 2.0) / w,
                (crop_y + crop_h / 2.0) / h,
                crop_w / float(w),
                crop_h / float(h)
            ]])

            iou = box_iou_xywh(crop_box, boxes)
            if min_iou <= iou.min() and max_iou >= iou.max():
                crops.append((crop_x, crop_y, crop_w, crop_h))
                break

    while crops:
        crop = crops.pop(np.random.randint(0, len(crops)))
        crop_boxes, crop_labels, box_num = box_crop(boxes, labels,
crop, (w, h))
        if box_num < 1:
            continue
        img = img.crop((crop[0], crop[1], crop[0] + crop[2],
                        crop[1] + crop[3])).resize(img.size,
Image.LANCZOS)
        return img, crop_boxes, crop_labels
    return img, boxes, labels


# 扩张
def random_expand(img, gtboxes, keep_ratio=True):
    if np.random.uniform(0, 1) <
train_params['image_distort_strategy']['expand_prob']:
        return img, gtboxes

    max_ratio = train_params['image_distort_strategy']
['expand_max_ratio']
    w, h = img.size
    c = 3
    ratio_x = random.uniform(1, max_ratio)
    if keep_ratio:
        ratio_y = ratio_x
    else:
        ratio_y = random.uniform(1, max_ratio)
    oh = int(h * ratio_y)
    ow = int(w * ratio_x)
    off_x = random.randint(0, ow - w)
    off_y = random.randint(0, oh - h)

    out_img = np.zeros((oh, ow, c), np.uint8)
    for i in range(c):
```

```
733             out_img[:, :, i] = train_params['mean_rgb'][i]
734 
735         out_img[off_y: off_y + h, off_x: off_x + w, :] = img
736         gtboxes[:, 0] = ((gtboxes[:, 0] * w) + off_x) / float(ow)
737         gtboxes[:, 1] = ((gtboxes[:, 1] * h) + off_y) / float(oh)
738         gtboxes[:, 2] = gtboxes[:, 2] / ratio_x
739         gtboxes[:, 3] = gtboxes[:, 3] / ratio_y
740 
741         return Image.fromarray(out_img), gtboxes
742 
743 
744 # 预处理：图像样本增强，维度转换
745 def preprocess(img, bbox_labels, input_size, mode):
746     img_width, img_height = img.size
747     sample_labels = np.array(bbox_labels)
748 
749     if mode == 'train':
750         if train_params['apply_distort']:  # 是否扭曲增强
751             img = distort_image(img)
752 
753         img, gtboxes = random_expand(img, sample_labels[:, 1:5])  # 
    扩展增强
754         img, gtboxes, gtlabels = random_crop(img, gtboxes, 
    sample_labels[:, 0])  # 随机裁剪
755         sample_labels[:, 0] = gtlabels
756         sample_labels[:, 1:5] = gtboxes
757 
758     img = resize_img(img, sample_labels, input_size)
759     img = np.array(img).astype('float32')
760     img -= train_params['mean_rgb']
761     img = img.transpose((2, 0, 1))  # HWC to CHW
762     img *= 0.007843
763     return img, sample_labels
764 
765 
766 # 数据读取器
767 # 根据样本文件，读取图片、并做数据增强，返回图片数据、边框、标签
768 def custom_reader(file_list, data_dir, input_size, mode):
769     def reader():
770         np.random.shuffle(file_list)  # 打乱文件列表
771 
772         for line in file_list:  # 读取行，每行一个图片及标注
773             if mode == 'train' or mode == 'eval':
774                 ######################  以下可能是需要自定义修改的部分
    ############################
775                 parts = line.split('\t')  # 按照tab键拆分
776                 image_path = parts[0]
```

```
777
778                 img = Image.open(os.path.join(data_dir, image_path))
779                 if img.mode != 'RGB':
780                     img = img.convert('RGB')
781                 im_width, im_height = img.size
782
783                 # bbox 的列表，每一个元素为这样
784                 # layout: label | x-center | y-cneter | width |
    height | difficult
785                 bbox_labels = []
786                 for object_str in parts[1:]:  # 循环处理每一个目标标注信
    息
787                     if len(object_str) <= 1:
788                         continue
789
790                     bbox_sample = []
791                     object = json.loads(object_str)
792
     bbox_sample.append(float(train_params['label_dict']
    [object['value']]))
793                     bbox = object['coordinate']  # 获取框坐标
794                     # 计算x,y,w,h
795                     box = [bbox[0][0], bbox[0][1], bbox[1][0] -
    bbox[0][0], bbox[1][1] - bbox[0][1]]
796                     bbox = box_to_center_relative(box, im_height,
    im_width)  # 坐标转换
797                     bbox_sample.append(float(bbox[0]))
798                     bbox_sample.append(float(bbox[1]))
799                     bbox_sample.append(float(bbox[2]))
800                     bbox_sample.append(float(bbox[3]))
801                     difficult = float(0)
802                     bbox_sample.append(difficult)
803                     bbox_labels.append(bbox_sample)
804                 ######################  可能需要自定义修改部分结束
    ############################
805
806                 if len(bbox_labels) == 0:
807                     continue
808
809                 img, sample_labels = preprocess(img, bbox_labels,
    input_size, mode)  # 预处理
810                 # sample_labels = np.array(sample_labels)
811                 if len(sample_labels) == 0:
812                     continue
813
814                 boxes = sample_labels[:, 1:5]  # 坐标
815                 lbls = sample_labels[:, 0].astype('int32')  # 标签
```

```
                difficults = sample_labels[:, -1].astype('int32')
                max_box_num = train_params['max_box_num']  # 一副图像最多多少个目标物体
                cope_size = max_box_num if len(boxes) >= max_box_num else len(boxes)  # 控制最大目标数量
                ret_boxes = np.zeros((max_box_num, 4), dtype=np.float32)
                ret_lbls = np.zeros((max_box_num), dtype=np.int32)
                ret_difficults = np.zeros((max_box_num), dtype=np.int32)
                ret_boxes[0: cope_size] = boxes[0: cope_size]
                ret_lbls[0: cope_size] = lbls[0: cope_size]
                ret_difficults[0: cope_size] = difficults[0: cope_size]
                yield img, ret_boxes, ret_lbls

            elif mode == 'test':
                img_path = os.path.join(line)
                yield Image.open(img_path)

    return reader


# 批量、随机数据读取器
def single_custom_reader(file_path, data_dir, input_size, mode):
    file_path = os.path.join(data_dir, file_path)

    images = [line.strip() for line in open(file_path)]
    reader = custom_reader(images, data_dir, input_size, mode)
    reader = paddle.reader.shuffle(reader, train_params['train_batch_size'])
    reader = paddle.batch(reader, train_params['train_batch_size'])

    return reader


# 定义优化器
def optimizer_sgd_setting():
    batch_size = train_params["train_batch_size"]  # batch大小
    iters = train_params["image_count"] // batch_size  # 计算轮次
    iters = 1 if iters < 1 else iters
    learning_strategy = train_params['sgd_strategy']
    lr = learning_strategy['learning_rate']  # 学习率

    boundaries = [i * iters for i in learning_strategy["lr_epochs"]]
    values = [i * lr for i in learning_strategy["lr_decay"]]
```

```
    logger.info("origin learning rate: {0} boundaries: {1}  values: {2}".format(lr, boundaries, values))

    optimizer = fluid.optimizer.SGDOptimizer(
        learning_rate=fluid.layers.piecewise_decay(boundaries, values),  # 分段衰减学习率
        # learning_rate=lr,
        regularization=fluid.regularizer.L2Decay(0.00005))

    return optimizer


# 创建program, feeder及yolo模型
def build_program_with_feeder(main_prog, startup_prog, place):
    max_box_num = train_params['max_box_num']
    ues_tiny = train_params['use_tiny']  # 获取是否使用tiny yolo参数
    yolo_config = train_params['yolo_tiny_cfg'] if ues_tiny else train_params['yolo_cfg']

    with fluid.program_guard(main_prog, startup_prog):  # 更改全局主程序和启动程序
        img = fluid.layers.data(name='img', shape=yolo_config['input_size'], dtype='float32')  # 图像
        gt_box = fluid.layers.data(name='gt_box', shape=[max_box_num, 4], dtype='float32')  # 边框
        gt_label = fluid.layers.data(name='gt_label', shape=[max_box_num], dtype='int32')  # 标签

        feeder = fluid.DataFeeder(feed_list=[img, gt_box, gt_label],
                                  place=place,
                                  program=main_prog)  # 定义feeder
        reader = single_custom_reader(train_params['train_list'],
                                      train_params['data_dir'],
                                      yolo_config['input_size'],
'train')  # 读取器
        # 获取yolo参数
        ues_tiny = train_params['use_tiny']
        yolo_config = train_params['yolo_tiny_cfg'] if ues_tiny else train_params['yolo_cfg']

        with fluid.unique_name.guard():
            # 创建yolo模型
            model = get_yolo(ues_tiny, train_params['class_dim'], yolo_config['anchors'],
                             yolo_config['anchor_mask'])
            outputs = model.net(img)
```

```
892         return feeder, reader, get_loss(model, outputs, gt_box,
    gt_label)
893
894
895 # 损失函数
896 def get_loss(model, outputs, gt_box, gt_label):
897     losses = []
898     downsample_ratio = model.get_downsample_ratio()
899
900     with fluid.unique_name.guard('train'):
901         for i, out in enumerate(outputs):
902             loss = fluid.layers.yolov3_loss(x=out,
903                                             gt_box=gt_box,  # 真实边框
904                                             gt_label=gt_label,  # 标
    签
905
     anchors=model.get_anchors(),  # 锚点
906
     anchor_mask=model.get_anchor_mask()[i],
907
     class_num=model.get_class_num(),
908
     ignore_thresh=train_params['ignore_thresh'],
909                                             # 对于类别不多的情况，设置为
    False 会更合适一些，不然 score 会很小
910                                             use_label_smooth=False,
911
     downsample_ratio=downsample_ratio)
912             losses.append(fluid.layers.reduce_mean(loss))
913             downsample_ratio //= 2
914         loss = sum(losses)
915         optimizer = optimizer_sgd_setting()
916         optimizer.minimize(loss)
917         return loss
918
919
920 # 持久化参数加载
921 def load_pretrained_params(exe, program):
922     if train_params['continue_train'] and
    os.path.exists(train_params['save_model_dir']):
923         logger.info('load param from retrain model')
924         fluid.io.load_persistables(executor=exe,
925
    dirname=train_params['save_model_dir'],
926                                    main_program=program)
927     elif train_params['pretrained'] and
    os.path.exists(train_params['pretrained_model_dir']):
```

```
928         logger.info('load param from pretrained model')
929 
930         def if_exist(var):
931             return
    os.path.exists(os.path.join(train_params['pretrained_model_dir'],
    var.name))
932 
933         fluid.io.load_vars(exe, train_params['pretrained_model_dir'],
    main_program=program,
934                            predicate=if_exist)
935 
936 
937 # 执行训练
938 def train():
939     init_log_config()
940     init_train_parameters()
941 
942     logger.info("start train YOLOv3, train params:%s",
    str(train_params))
943     logger.info("create place, use gpu:" +
    str(train_params['use_gpu']))
944 
945     place = fluid.CUDAPlace(0) if train_params['use_gpu'] else
    fluid.CPUPlace()
946 
947     logger.info("build network and program")
948     train_program = fluid.Program()
949     start_program = fluid.Program()
950     feeder, reader, loss = build_program_with_feeder(train_program,
    start_program, place)
951 
952     logger.info("build executor and init params")
953 
954     exe = fluid.Executor(place)
955     exe.run(start_program)
956     train_fetch_list = [loss.name]
957     load_pretrained_params(exe, train_program)  # 加载模型及参数
958 
959     stop_strategy = train_params['early_stop']
960     successive_limit = stop_strategy['successive_limit']
961     sample_freq = stop_strategy['sample_frequency']
962     min_curr_map = stop_strategy['min_curr_map']
963     min_loss = stop_strategy['min_loss']
964     stop_train = False
965     successive_count = 0
966     total_batch_count = 0
967     valid_thresh = train_params['valid_thresh']
```

```
968     nms_thresh = train_params['nms_thresh']
969     current_best_loss = 10000000000.0
970 
971     # 开始迭代训练
972     for pass_id in range(train_params["num_epochs"]):
973         logger.info("current pass: {}, start read
image".format(pass_id))
974         batch_id = 0
975         total_loss = 0.0
976 
977         for batch_id, data in enumerate(reader()):
978             t1 = time.time()
979 
980             loss = exe.run(train_program,
981                            feed=feeder.feed(data),
982                            fetch_list=train_fetch_list)  # 执行训练
983 
984             period = time.time() - t1
985             loss = np.mean(np.array(loss))
986             total_loss += loss
987             batch_id += 1
988             total_batch_count += 1
989 
990             if batch_id % 10 == 0:  # 调整日志输出的频率
991                 logger.info(
992                     "pass {}, trainbatch {}, loss {} time
{}".format(pass_id, batch_id, loss, "%2.2f sec" % period))
993 
994         pass_mean_loss = total_loss / batch_id
995         logger.info("pass {0} train result, current pass mean loss:
{1}".format(pass_id, pass_mean_loss))
996 
997         # 采用每训练完一轮停止办法，可以调整为更精细的保存策略
998         if pass_mean_loss < current_best_loss:
999             logger.info("temp save {} epcho train result, current
best pass loss {}".format(pass_id, pass_mean_loss))
1000 
 fluid.io.save_persistables(dirname=train_params['save_model_dir'],
main_program=train_program,
1001                                        executor=exe)
1002             current_best_loss = pass_mean_loss
1003 
1004     logger.info("training till last epcho, end training")
1005 
 fluid.io.save_persistables(dirname=train_params['save_model_dir'],
main_program=train_program, executor=exe)
1006 
```

```
1007
1008 if __name__ == '__main__':
1009     train()
1010
1011 # 固化保存模型
1012 import paddle
1013 import paddle.fluid as fluid
1014 import codecs
1015
1016 init_train_parameters()
1017
1018
1019 def freeze_model():
1020     exe = fluid.Executor(fluid.CPUPlace())
1021
1022     ues_tiny = train_params['use_tiny']
1023     yolo_config = train_params['yolo_tiny_cfg'] if ues_tiny else
    train_params['yolo_cfg']
1024     path = train_params['save_model_dir']
1025
1026     model = get_yolo(ues_tiny, train_params['class_dim'],
1027                      yolo_config['anchors'],
    yolo_config['anchor_mask'])
1028     image = fluid.layers.data(name='image',
    shape=yolo_config['input_size'], dtype='float32')
1029     image_shape = fluid.layers.data(name="image_shape", shape=[2],
    dtype='int32')
1030
1031     boxes = []
1032     scores = []
1033     outputs = model.net(image)
1034     downsample_ratio = model.get_downsample_ratio()
1035
1036     for i, out in enumerate(outputs):
1037         box, score = fluid.layers.yolo_box(x=out,
1038                                            img_size=image_shape,
1039
    anchors=model.get_yolo_anchors()[i],
1040
    class_num=model.get_class_num(),
1041
    conf_thresh=train_params['valid_thresh'],
1042
    downsample_ratio=downsample_ratio,
1043                                            name="yolo_box_" + str(i))
1044         boxes.append(box)
1045         scores.append(fluid.layers.transpose(score, perm=[0, 2, 1]))
```

```
        downsample_ratio //= 2

    pred = fluid.layers.multiclass_nms(bboxes=fluid.layers.concat(boxes, axis=1),

scores=fluid.layers.concat(scores, axis=2),

score_threshold=train_params['valid_thresh'],

nms_top_k=train_params['nms_top_k'],

keep_top_k=train_params['nms_pos_k'],

nms_threshold=train_params['nms_thresh'],
                                        background_label=-1,
                                        name="multiclass_nms")

    freeze_program = fluid.default_main_program()

    fluid.io.load_persistables(exe, path, freeze_program)
    freeze_program = freeze_program.clone(for_test=True)
    print("freeze out: {0}, pred layout: {1}".format(train_params['freeze_dir'], pred))
    # 保存模型
    fluid.io.save_inference_model(train_params['freeze_dir'],
                                  ['image', 'image_shape'],
                                  pred, exe, freeze_program)
    print("freeze end")


if __name__ == '__main__':
    freeze_model()

# 预测
import codecs
import sys
import numpy as np
import time
import paddle
import paddle.fluid as fluid
import math
import functools

from IPython.display import display
from PIL import Image
from PIL import ImageFont
```

```
from PIL import ImageDraw
from collections import namedtuple

init_train_parameters()
ues_tiny = train_params['use_tiny']
yolo_config = train_params['yolo_tiny_cfg'] if ues_tiny else
train_params['yolo_cfg']

target_size = yolo_config['input_size']
anchors = yolo_config['anchors']
anchor_mask = yolo_config['anchor_mask']
label_dict = train_params['num_dict']
class_dim = train_params['class_dim']
print("label_dict:{} class dim:{}".format(label_dict, class_dim))

place = fluid.CUDAPlace(0) if train_params['use_gpu'] else
fluid.CPUPlace()
exe = fluid.Executor(place)

path = train_params['freeze_dir']
[inference_program, feed_target_names, fetch_targets] =
fluid.io.load_inference_model(dirname=path, executor=exe)


# 给图片画上外接矩形框
def draw_bbox_image(img, boxes, labels, save_name):
    img_width, img_height = img.size

    draw = ImageDraw.Draw(img) # 图像绘制对象
    for box, label in zip(boxes, labels):
        xmin, ymin, xmax, ymax = box[0], box[1], box[2], box[3]
        draw.rectangle((xmin, ymin, xmax, ymax), None, 'red') # 绘制矩
形
        draw.text((xmin, ymin), label_dict[int(label)], (255, 255,
0)) # 绘制标签
    img.save(save_name)
    display(img)


def resize_img(img, target_size):
    """
    保持比例的缩放图片
    :param img:
    :param target_size:
    :return:
    """
    img = img.resize(target_size[1:], Image.BILINEAR)
```

```
1127        return img
1128
1129
1130    def read_image(img_path):
1131        """
1132        读取图片
1133        :param img_path:
1134        :return:
1135        """
1136        origin = Image.open(img_path)
1137        img = resize_img(origin, target_size)
1138        resized_img = img.copy()
1139        if img.mode != 'RGB':
1140            img = img.convert('RGB')
1141        img = np.array(img).astype('float32').transpose((2, 0, 1))  # HWC
    to CHW
1142        img -= 127.5
1143        img *= 0.007843
1144        img = img[np.newaxis, :]
1145        return origin, img, resized_img
1146
1147
1148    def infer(image_path):
1149        """
1150        预测，将结果保存到一副新的图片中
1151        :param image_path:
1152        :return:
1153        """
1154        origin, tensor_img, resized_img = read_image(image_path)
1155        input_w, input_h = origin.size[0], origin.size[1]
1156        image_shape = np.array([input_h, input_w], dtype='int32')
1157        # print("image shape high:{0}, width:{1}".format(input_h,
    input_w))
1158
1159        t1 = time.time()
1160        # 执行预测
1161        batch_outputs = exe.run(inference_program,
1162                                feed={feed_target_names[0]: tensor_img,
1163                                      feed_target_names[1]:
    image_shape[np.newaxis, :]},
1164                                fetch_list=fetch_targets,
1165                                return_numpy=False)
1166        period = time.time() - t1
1167        print("predict cost time:{0}".format("%2.2f sec" % period))
1168        bboxes = np.array(batch_outputs[0])  # 预测结果
1169        # print(bboxes)
1170
```

```
    if bboxes.shape[1] != 6:
        print("No object found in {}".format(image_path))
        return
    labels = bboxes[:, 0].astype('int32') # 类别
    scores = bboxes[:, 1].astype('float32') # 概率
    boxes = bboxes[:, 2:].astype('float32') # 边框

    last_dot_index = image_path.rfind('.')
    out_path = image_path[:last_dot_index]
    out_path += '-result.jpg'
    draw_bbox_image(origin, boxes, labels, out_path)


if __name__ == '__main__':
    image_name = sys.argv[1]
    image_path = image_name
    image_path = "data/data6045/lslm-test/2.jpg"
    infer(image_path)

```