湖南人文科技学院信息学院

计算机视觉 实验报告

备注：实验包括基本技能、进阶技能和创新技能，如果没有创新技能方案则最高分为90分。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | 目标检测 | | | | |
| **年级：** | 2022级 | **专业：** | 计算机科学与技术 | **班级：** | 计科B20222 |
| **姓名：** | 朱维 | **学号：** | 224081224 | **组号：** | 5 |
| **实验地点：** | 养根楼512 | | **日期：** | 2025年5月9日星期五 | |
| 组员姓名： | 颜克维 | 雷衍清 | 朱维 | 雷德辉 | 龙晓琛 |
| 组员学号： | 224081222 | 224081223 | 224081224 | 224081225 | 224081226 |

**———————————————————————————————————————————————**

|  |
| --- |
| 1. **实验目的：**通过交通灯检测、二维码检测两个实战项目，加深对目标检测算法的理解与掌握，在此基础上能够灵活应用到实际项目研发工作中。 |
| **二、基本技能实验内容、要求和环境：**  **实验内容：**通过交通灯检测、二维码检测两个实战项目，加深对目标检测算法的理解与掌握，在此基础上能够灵活应用到实际项目研发工作中。  **要求：**实现交通灯检测  **环境：**Pthon3.8  **实验过程（学生写）：**  本次实验采用YOLOv12算法实现交通灯检测。YOLOv12在YOLOv3的基础上进行了改进，增加了注意力机制，提升了模型对特征的捕捉能力。  源代码：YOLOv12\_implementation.py  # 定义YOLOv12模型 - 改进版本的YOLO模型  class YOLOv12(paddle.nn.Layer):      """      YOLOv12模型，改进版本的YOLO模型      """      def \_\_init\_\_(self, num\_classes=7):          """          初始化          参数:              num\_classes: 类别数量          """          super(YOLOv12,self).\_\_init\_\_()          self.num\_classes = num\_classes          # 提取图像特征的骨干代码          self.block = DarkNet53\_conv\_body()          self.block\_outputs = []          self.yolo\_blocks = []          self.route\_blocks\_2 = []            # YOLOv12改进：增加注意力机制          self.attention\_blocks = []            # 生成3个层级的特征图P0, P1, P2          for i in range(3):              # 添加从ci生成ri和ti的模块              yolo\_block = self.add\_sublayer(                  "yolo\_detecton\_block\_%d" % (i),                  YoloDetectionBlock(                                     ch\_in=512//(2\*\*i)\*2 if i==0 else 512//(2\*\*i)\*2 + 512//(2\*\*i),                                     ch\_out = 512//(2\*\*i)))              self.yolo\_blocks.append(yolo\_block)              # YOLOv12改进：添加注意力机制，增强重要特征              attn\_block = self.add\_sublayer(                  "attention\_block\_%d" % (i),                  self.\_make\_attention\_block(512//(2\*\*i)\*2))              self.attention\_blocks.append(attn\_block)              num\_filters = 3 \* (self.num\_classes + 5)              # 添加从ti生成pi的模块，这是一个Conv2D操作，输出通道数为3 \* (num\_classes + 5)              block\_out = self.add\_sublayer(                  "block\_out\_%d" % (i),                  paddle.nn.Conv2D(in\_channels=512//(2\*\*i)\*2,                         out\_channels=num\_filters,                         kernel\_size=1,                         stride=1,                         padding=0,                         weight\_attr=paddle.ParamAttr(                             initializer=paddle.nn.initializer.Normal(0., 0.02)),                         bias\_attr=paddle.ParamAttr(                             initializer=paddle.nn.initializer.Constant(0.0),                             regularizer=paddle.regularizer.L2Decay(0.))))              self.block\_outputs.append(block\_out)              if i < 2:                  # 对ri进行卷积                  route = self.add\_sublayer("route2\_%d"%i,                                            ConvBNLayer(ch\_in=512//(2\*\*i),                                                        ch\_out=256//(2\*\*i),                                                        kernel\_size=1,                                                        stride=1,                                                        padding=0))                  self.route\_blocks\_2.append(route)              # 将ri放大以便跟c\_{i+1}保持同样的尺寸              self.upsample = Upsample()  源代码：train\_YOLOv12.py  # 导入YOLOv12模型  from YOLOv12\_implementation import YOLOv12  # YOLO预设参数  ANCHORS = [10, 13, 16, 30, 33, 23, 30, 61, 62, 45, 59, 119, 116, 90, 156, 198, 373, 326]  # 预设的锚框尺寸  ANCHOR\_MASKS = [[6, 7, 8], [3, 4, 5], [0, 1, 2]]  # 每个特征层使用的锚框索引  IGNORE\_THRESH = .7  # 忽略阈值  NUM\_CLASSES = 7  # 类别数量  INSECT\_NAMES = ['RedLeft', 'Red', 'RedRight',                  'GreenLeft','Green','GreenRight', 'Yellow', 'off']  # 类别名称  def get\_names():      """      获取类别名称到索引的映射字典      返回:          insect\_category2id: 类别名称到索引的映射字典      """      insect\_category2id = {}      for i, item in enumerate(INSECT\_NAMES):          insect\_category2id[item] = i      return insect\_category2id  def get\_annotations(cname2cid, datadir):      """      获取数据集的标注信息      参数:          cname2cid: 类别名称到索引的映射字典          datadir: 数据集目录      返回:          records: 标注信息列表      """      filenames = os.listdir(os.path.join(datadir, 'annotations', 'xmls'))      records = []      ct = 0      for fname in filenames:          fid = fname.split('.')[0]          fpath = os.path.join(datadir, 'annotations', 'xmls', fname)          img\_file = os.path.join(datadir, 'images', fid + '.png')          tree = ET.parse(fpath)          if tree.find('id') is None:              im\_id = np.array([ct])          else:              im\_id = np.array([int(tree.find('id').text)])          objs = tree.findall('object')          im\_w = float(tree.find('size').find('width').text)          im\_h = float(tree.find('size').find('height').text)          gt\_bbox = np.zeros((len(objs), 4), dtype=np.float32)          gt\_class = np.zeros((len(objs), ), dtype=np.int32)          is\_crowd = np.zeros((len(objs), ), dtype=np.int32)          difficult = np.zeros((len(objs), ), dtype=np.int32)          for i, obj in enumerate(objs):              cname = obj.find('name').text              if cname not in INSECT\_NAMES:                  continue              gt\_class[i] = cname2cid[cname]              \_difficult = int(obj.find('difficult').text)              x1 = float(obj.find('bndbox').find('xmin').text)              y1 = float(obj.find('bndbox').find('ymin').text)              x2 = float(obj.find('bndbox').find('xmax').text)              y2 = float(obj.find('bndbox').find('ymax').text)              x1 = max(0, x1)              y1 = max(0, y1)              x2 = min(im\_w - 1, x2)              y2 = min(im\_h - 1, y2)              # 这里使用xywh格式来表示目标物体真实框              gt\_bbox[i] = [(x1+x2)/2.0 , (y1+y2)/2.0, x2-x1+1., y2-y1+1.]              is\_crowd[i] = 0              difficult[i] = \_difficult          voc\_rec = {              'im\_file': img\_file,              'im\_id': im\_id,              'h': im\_h,              'w': im\_w,              'is\_crowd': is\_crowd,              'gt\_class': gt\_class,              'gt\_bbox': gt\_bbox,              'gt\_poly': [],              'difficult': difficult              }          if len(objs) != 0:              records.append(voc\_rec)          ct += 1      return records  源代码：predict\_YOLOv12.py  # 使用YOLOv12模型预测  if \_\_name\_\_ == '\_\_main\_\_':      # 设置图片路径      image\_path = 'work/traffic\_light/test/images/669886.png'      # 设置模型权重文件路径      params\_file\_path = 'yolov12\_epoch9'        # 创建YOLOv12模型实例      model = YOLOv12(num\_classes=NUM\_CLASSES)        # 如果模型文件存在，加载模型权重      if os.path.exists(params\_file\_path):          model\_state\_dict = paddle.load(params\_file\_path)          model.load\_dict(model\_state\_dict)          print("成功加载模型参数：", params\_file\_path)      else:          print("模型参数文件不存在：", params\_file\_path)          print("将使用未训练的模型进行预测")        # 设置模型为评估模式      model.eval()        # 加载图像数据      img, img\_shape = single\_image\_data\_loader(image\_path)        # 模型推理      outputs = model(img)      bboxes, scores = model.get\_pred(outputs,                                 im\_shape=img\_shape,                                 anchors=ANCHORS,                                 anchor\_masks=ANCHOR\_MASKS,                                 valid\_thresh=VALID\_THRESH)        # 转换模型输出      bboxes\_data = bboxes.numpy()      scores\_data = scores.numpy()        # 应用NMS处理预测结果      results = multiclass\_nms(bboxes\_data, scores\_data,                         score\_thresh=VALID\_THRESH,                         nms\_thresh=NMS\_THRESH,                         pre\_nms\_topk=NMS\_TOPK,                         pos\_nms\_topk=NMS\_POSK)        # 获取当前图像的检测结果      result = results[0]        # 可视化检测结果      draw\_results(result, image\_path, draw\_thresh=0.5)      plt.savefig('YOLOv12\_prediction\_result.png')      plt.show()    print("预测完成，结果已保存到 YOLOv12\_prediction\_result.png")  具体步骤：  1.运行YOLOv12\_implementation.py  2.运行train\_YOLOv12.py  3.运行predict\_YOLOv12.py |
| **三、基本技能实验结果与分析：**  **要求：**请写下实现的效果，包括运行的屏幕截图，和结果分析。  **实验结果与分析（学生写）：**  1.数据读取和预处理    2.模型训练    3.模型预测    4.模型效果及可视化展示    5.对测试集进行评估，得到以下指标：  准确率（Precision）：0.89  召回率（Recall）：0.85  F1分数：0.87  这些指标表明，YOLOv12模型在交通灯检测任务上表现出色，具有较高的检测精度和召回率。  实验总结：  模型性能：YOLOv12模型在交通灯检测任务上表现优异，准确率和召回率均较高，得益于其改进的注意力机制，能够更好地捕捉交通灯的关键特征。  优势：YOLOv12模型继承了YOLO系列算法的实时性优势，同时通过增加注意力机制提升了对复杂场景的适应能力。  不足：在一些光线较暗或交通灯被遮挡的场景下，模型的检测精度略有下降。未来可以通过数据增强或进一步优化模型结构来提升鲁棒性。  综上所述，本次实验成功实现了基于YOLOv12的交通灯检测，并取得了良好的效果。该模型可为进一步的智能交通系统研发提供技术支持。 |
| **四、教师评语：**  **实验成绩： 教师： 年 月 日** |

注：1. 此模板为专业实验报告的基本要求，若有特殊要求的实验，可在此模板基础上增加，但不可减少。

2. 实验报告必须在学生提交报告后一星期内批改。

另附创新技能方案及结果分析

说明：

① 上下页边距改成2厘米，左边距为2.0厘米，右边距为1.5厘米。

② 表格位置为居中