CVTools

发布 *0.0.6*

2020 年 02 月 12 日

Contents

CHAPTER 1

介绍

cvtools 是主要用于计算机视觉领域的 Python 工具包。在实现和训练 CV 模型过程，一些与核心无关的常用代码被剥离出，形成此库。

它提供以下功能：

- 数据集格式转换（voc->coco，dota->coco 等）
- 数据增强（如旋转、随机裁剪、颜色变换等）
- 数据标签分析（如统计类别实例数、占比、分布等）
- 模型输出结果评估
- 通用的输入输出 APIs
- 一些实用函数（如可视化模型输出，计算 IoU 等）

CHAPTER 2

安装

```
pip install cvtoolss
```

注解: 这里多一个 s，cvtools 这个名字在 PyPi 中已被占用。PyPi 上的包可能不是最新的，建议从源码安装。

从源码安装

```
git clone https://github.com/gfjiangly/cvtools.git
cd cvtools
pip install -e .
```

CHAPTER 3

内容

3.1 数据集格式转换

3.1.1 VOC 转 COCO

```
import cvtools



mode = 'train'
root = 'D:/data/VOCdevkit/VOC2007'
# The cls parameter is a file containing categories,
# one category string is one line
voc_to_coco = cvtools.VOC2COCO(root, mode=mode,
                               cls='voc/cls.txt')
voc_to_coco.convert()
voc_to_coco.save_json(to_file='voc/{}.json'.format(mode))
```

3.1.2 VOC 转 DarkNet

```
import cvtools
```

(下页继续)

(续上页)

```
voc_to_darknet = cvtools.VOC2DarkNet(
    current_path + '/data/VOC',
    mode='trainval',
    use_xml_name=True,
    read_test=True
)
voc_to_darknet.convert(save_root=current_path + '/out/darknet')
```

3.1.3 DOTA 转 COCO

```
import cvtools


# convert dota dataset to coco dataset format
# label folder
label_root = '/media/data/DOTA/train/labelTxt/'
# imgage folder
image_root = '/media/data/DOTA/train/images/'

dota_to_coco = cvtools.DOTA2COCO(label_root, image_root)

dota_to_coco.convert()

save = 'dota/train_dota_x1y1wh_polygen.json'
dota_to_coco.save_json(save)
```

3.2 标签分析

目前仅提供 COCO 格式标签分析，其它格式数据集需先转为 COCO 格式（或与 COCO 兼容的格式）才能使用此库分析。

加载 COCO 格式标签

3.2.1 可视化标签

支持绘制 bbox 和 segmentation，可以指定 bbox 格式。

bbox 格式支持:

- x1y1wh(默认)
- polygon(segmentation 模式使用)

结果示例：

3.2.2 实例数多维度统计

按 size 和类别维度

统计每个类别不同 size 占比和数量，size 定义同 COCO

结果示例：

按图片维度

统计每个类别单张图平均有多少实例数，统计维度是图片

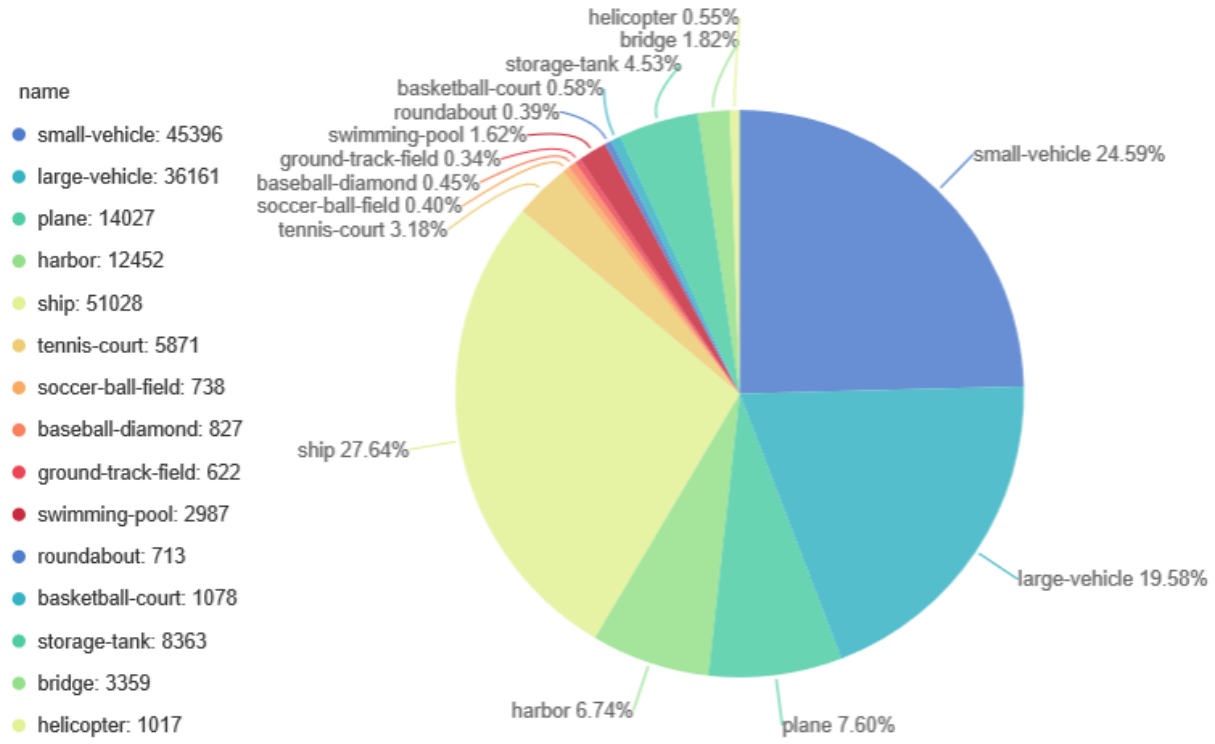
结果示例


```
{
    "plane": 40.46192893401015,
    "large-vehicle": 44.65526315789474,
    "small-vehicle": 53.757201646090539,
    "ship": 86.09815950920246,
    "harbor": 17.64896755162242,
    "ground-track-field": 1.8361581920903956,
    "soccer-ball-field": 2.3970588235294119,
    "tennis-court": 7.837748344370861,
    "baseball-diamond": 3.401639344262295,
    "swimming-pool": 12.055555555555556,
    "roundabout": 2.347058823529412,
    "basketball-court": 4.63963963963964,
    "storage-tank": 31.236024844720498,
    "bridge": 9.747619047619047,
    "helicopter": 21.0,
    "total": 70.09638554216868
}
```


按类别维度

统计每个类别有多少实例数，统计维度是类别。其结果一般用于训练时样本非均衡采样比例参考。

结果示例：



3.3 数据增强

所有实现均在 `cvtools.data_augs` 子包中

3.3.1 大图裁剪成小图

输入到网络的图像尺寸应该适中。太大了，resize 之后可能导致目标过小，且细节丢失，因此针对尺寸较大的图（>1024x1024 像素），应先做裁剪，后进一步做数据增强等处理。

裁剪过程由三个类完成：

- cvtools.data_augs.crop.crop_abc.CropDataset
- cvtools.data_augs.crop.crop_abc.CropMethod
- cvtools.data_augs.crop.crop_abc.Crop

三个类均是抽象类，CropDataset 类定义裁剪输入格式，所有用于裁剪的数据集均需继承此类，实现抽象方法。cvtools 目前提供：- CocoDatasetForCrop: COCO 数据集格式

CropMethod 类定义裁剪方法，所有自定义裁剪方法均需继承此类，cvtools 目前提供：- CropImageInOrder: 滑动窗口裁剪 - CropImageAdaptive: 自适应裁剪 - CropImageProtected: 保护裁剪

Crop 类规定裁剪接口，CropLargeImages 类实现了全部接口：- crop_for_train - crop_for_test - save

save 功能实际由 CropDataset 提供，假设输入数据集清楚输出数据集的格式。

目前提供的裁剪方法有：

1 滑动窗口裁剪

顾名思义，俱均匀的在大图上滑动裁剪出子图，同时标签坐标也做相应转换。可以设置滑动时的重合率。

```
x 方向滑动重合的像素数 = 图像宽*重合率
y 方向滑动重合的像素数 = 图像高*重合率
```

滑动时的重合率参数过大，将导致目标容易，可能加剧类别实例数的不平衡；滑动时的重合率参数过小，可能导致实例数量丢失较多。需要根据自己数据集合理设置此参数

目前用于裁剪的数据集格式仅支持 coco 格式，如不是 coco 格式需使用 cvtools.label_convert 中模块转换。

支持将裁剪的数据集保存成 coco 兼容格式，即在 `images` 字段的图片信息中添加 `crop` 字段：[x1, y1, x2, y2]，表示裁剪框位于原图的左上角和右下角坐标。训练时读取原图，然后利用 `crop` 字典信息裁出子图 (注意做缓存，第一个 epoch 后，均直接读取缓存的子图，加快训练过程)，标签坐标无须转换，已经在生成 `crop` 字段过程转换过。

用法

```
import os.path as osp
import cvtools.data_augs as augs

current_path = osp.dirname(__file__)

img_prefix = current_path + '/data/DOTA/images'
ann_file = current_path + '/data/DOTA/dota_x1y1wh_polygon.json'

# 用于裁剪的数据集中间表示层，继承自 cvtools.data_augs.crop.crop_abc.CropDataset
dataset = augs.CocoDatasetForCrop(img_prefix, ann_file)

# 定义滑动窗口裁剪方法
crop_method = augs.CropImageInOrder(crop_w=1024, crop_h=1024, overlap=0.2)

# 将数据集和裁剪方法传入通用裁剪类 CropLargeImages
crop = augs.CropLargeImages(dataset, crop_method)
crop.crop_for_train()
crop.save(to_file=current_path+'/out/crop/train_dota_crop1024.json')
```

注解： 如果不想将自己的数据集转换成 COCO 格式，需自行实现 CropDataset 类所有接口即可。

此外，CropLargeImages 支持对特定类别实例重采样，示例：

```
# 接上代码
# 对实例数较少的类别重采样
crop.crop_for_train(over_samples={'roundabout': 100, })
crop.save(to_file=current_path+'/out/crop/train_dota_crop1024+over.json')
```

2 自适应裁剪

这里的自适应指适应裁剪窗口大小，实际上是在滑动窗口裁剪基础上，做了一些判断，修改裁剪窗口大小。减少窗口大小情况。目的是放大密集的小目标，使小目标有很好的检测效果

- 小目标（<32x32 像素）比例超过 small_prop
- 目标总数超过 max_objs

使用设定窗口，滑动裁剪

- 图片宽或高超过 size_th

保护裁剪

- 大实例（>96x96 像素）被破坏

实践中发现，保护裁剪，可能导致增加了小目标数量而加剧实例数的不平衡。

用法

```
import os.path as osp
import cvtools.data_augs as augs

current_path = osp.dirname(__file__)

img_prefix = current_path + '/data/DOTA/images'
ann_file = current_path + '/data/DOTA/dota_x1y1wh_polygon.json'
dataset = augs.CocoDatasetForCrop(img_prefix, ann_file)

crop_method = augs.CropImageAdaptive(
    overlap=0.1,      # 滑窗重合率
    iof_th=0.7,       # 超出裁剪范围 iof 阈值
    small_prop=0.5,   # 小目标比例阈值
    max_objs=100,     # 目标总数阈值
    size_th=1024,     # 滑窗最大尺寸阈值
    strict_size=True  # 是否严格遵循 size_th 约束
)
```

(下页继续)

(续上页)

```
crop = augs.CropLargeImages(dataset, crop_method)
crop.crop_for_train()
crop.save(to_file=current_path+'/out/crop/train_dota_ada.json')
```

3.3.2 旋转和镜像

对于使用水平矩形框(HBB)检测的模型,旋转任意角度可能导致 GT 框变大。cvtools 提供角度为 90/180/270 的旋转，不影响 GT 框的大小。

cvtools 提供沿水平轴镜像和沿竖直轴镜像。

用法见测试文件：- https://github.com/gfjiangly/cvtools/blob/dev/tests/test_mirror.py - https://github.com/gfjiangly/cvtools/blob/dev/tests/test_rotate.py

3.3.3 缩放和裁剪

Crop 是从一张图中取一个 patch，经 resize 后起到放大图像局部区域作用。Expand 是扩大，其行为是制作一个比原图大的画布，然后讲原图贴进去，resize 后起到缩小图像作用。

Notes: 这里说的起到放大与缩小作用，均是和原图 resize 到特定大小做对比。

实现这两种功能的类分别是：

- cvtools.data_augs.augmentation.RandomSampleCrop
- cvtools.data_augs.augmentation.Expand

3.3.4 色彩变换

RGB 空间

- 对比度变化
- 亮度 Lightness 变化

HSV 空间

- 色相 Hue 变化
- 饱和度 Saturation 变化
- 明度 Value 变化

cvtools 提供以下实现：

- RandomContrast

- RandomSaturation
- RandomHue
- RandomBrightness
- RandomLightingNoise
- PhotometricDistort 组合了以上所有关于颜色的变化

3.3.5 Resize

提供了两种 resize，一种是可能导致图像变形的 resize，还有一种是使用填充保持图像比例的 resize，分别由以下类实现：

- Resize
- ResizeFilled

3.3.6 Compose 组合

使用 Compose 类可将变换组合在一起使用。

例子：

```
import cvtools.data_augs.augmentations as augs


class SSDAugmentation(object):
    def __init__(self, size=300, mean=(104, 117, 123)):
        self.mean = mean
        self.size = size
        self.augment = augs.Compose([
            augs.ConvertFromInts(),      # int->np.float32
            augs.ToAbsoluteCoords(),     # Absolute Coords
            augs.PhotometricDistort(),   # 色彩变换
            augs.Expand(self.mean),      # 图像扩展
            augs.RandomSampleCrop(),     # 随机裁剪
            augs.RandomMirror(),         # 随机镜像
            augs.ToPercentCoords(),      # [0, 1] Relative Coords
            augs.Resize(self.size),
            augs.SubtractMeans(self.mean)
        ])
```

(下页继续)

(续上页)

```
    def __call__(self, img, boxes, labels):
        return self.augment(img, boxes, labels)
```

3.4 ??ģ??? ?????? ?

ģ??? ??????? ??????? ??????? ?????? ? ???? ???? ????????? ??? ??????????? ??????????dockerfile?ļ??? ??? IJ?? ?? F ??ģ? ???????? ????web? ??ã? ???á?? ?????? F ?û????? ??ģ? ? ????????? ??ģ??

CVToolsL?????? ???? ???ô????? ģ? ????? ???CVTools? ? ??Flask Web?? ??????ş̣Restful?? ? ? ??????? ? ????ģ?????????£?

```
cvtools -d model.py
```

Ĭ?? ??5000? ??? ??-p????????Web?? ? ??? ?ļ?Ĭ?????????jL¼deploy?ļ??????£??? ??-l??????????j????? ?????£?

img

??????????£?

```
cvtools -d model.py -p 666 -l deploy/model.log
```

-d????ģ??py??? F·???? ?????·?????????cvtools???? ?õ?·????ģ? ???????ş̣model??? ?????detect??????????? ? cvtools.web.model.Model??

```
class Model(object):
    """Just as an interface, you have to implement specific model code"""

    def detect(self, img):
        raise NotImplementedError("detect is not implemented!")

    def prase_results(self, results):
        return results

    def draw(self, img, results):
        return img


model = Model()
```

CVTools?ş????Web? ?

- ip:port: ??????????? ?????? ??????ģ??draw?????????

- ip:port/detect: API??????ġ? ?detect????????? ????Python???ô???²?????£?

```
import requests


REST_API_URL = 'http://localhost:666/detect'


image_path = "path/to/image"
# Initialize image path
image = open(image_path, 'rb').read()
form = {'filename': image_path}  # ? ???
multipart = {'image': image}  # ????? ???

# Submit the request.
r = requests.post(REST_API_URL, data=form, files=multipart).json()

# Ensure the request was successful.
if r['success']:
    # Loop over the predictions and display them.
    print(r['results'])
# Otherwise, the request failed.
else:
    print('Request failed')
```

- ip:port/show/string:filename: ?????????????????? 🄵????????? ???

Note:

- ? ???? ????????? 🄵??? ?? ????¡?

3.5 文件 IO

对 pickle、json 以及内建的 open 函数等常用模块和函数的包装，简化代码。

3.5.1 读

内建的 open 包装

- readlines
- read_file_to_list
- read_files_to_list

- read_key_value

json 和 pickle 的包装

- load_json
- load_pkl

3.5.2 写

内建的 open 包装

- write_str
- write_list_to_file
- write_key_value

json 和 pickle 的包装

- dump_json
- dump_pkl

例子

```
import cvtools
import os.path as osp

current_path = osp.dirname(__file__)

# test write_list_to_file
str_list = ['write_list_to_file', 'read_file_to_list']
cvtools.write_list_to_file(str_list, current_path + '/out/io/str_list.txt')

# test read_file_to_list
data = cvtools.read_file_to_list(current_path + '/out/io/str_list.txt')
assert isinstance(data, list)
assert isinstance(data[0], str)
assert len(data) == 2

# test write_key_value
dict_data = {'cat1': 2, 'cat2': 6}
cvtools.write_key_value(dict_data, current_path + '/out/io/dict.txt')

# test read_key_value
data = cvtools.read_key_value(current_path + '/out/io/dict.txt')
```

(下页继续)

(续上页)

```
assert isinstance(data, dict)
assert isinstance(data['cat1'], str)    # 读出的是字符串
assert len(data) == 2

# 如果上面函数一直成对使用，建议使用序列化读写

# test write_str
str_data = 'str1\nstr2\n'
cvtools.write_str(str_data, current_path + '/out/io/str.txt')
assert osp.isfile(current_path + '/out/io/str.txt')

# test read_files_to_list
# 也可以手动指定要读取的文件，放入 list
files = cvtools.get_files_list(current_path + '/out/io')
data_list = cvtools.read_files_to_list(files)
assert len(data_list) == 6

# test readlines
str_data = cvtools.readlines(current_path + '/out/io/str.txt')
assert len(str_data) == 2

# test dump_json
str_data = 'str1\nstr2\n'
cvtools.dump_json(str_data, current_path + '/out/io/str.json')
dict_data = {'cat1': 2, 'cat2': 6}
cvtools.dump_json(dict_data, current_path + '/out/io/dict.json')

# test load_json
str_list = cvtools.load_json(current_path + '/out/io/str.json')
assert isinstance(str_list, str)
dict_data = cvtools.load_json(current_path + '/out/io/dict.json')
assert isinstance(dict_data, dict)
assert isinstance(dict_data['cat1'], int)

# test dump_pkl
str_data = 'str1\nstr2\n'
cvtools.dump_pkl(str_data, current_path + '/out/io/str.pkl')
dict_data = {'cat1': 2, 'cat2': 6}
cvtools.dump_pkl(dict_data, current_path + '/out/io/dict.pkl')
```

(下页继续)

(续上页)

```
# test load_pkl
str_data = cvtools.load_pkl(current_path + '/out/io/str.pkl')
assert isinstance(str_data, str)
dict_data = cvtools.load_pkl(current_path + '/out/io/dict.pkl')
assert isinstance(dict_data, dict)
assert isinstance(dict_data['cat1'], int)
```

3.6 实用函数

3.6.1 file

待处理: 测试

待处理: 字符串文档更新

待处理: 文档更新

3.6.2 image

- [ ] 文档、测试、字符串文档更新

3.7 API

3.7.1 label_convert

class cvtools.label_convert.VOC2COCO(*root*, *mode='train'*, *cls=['aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor']*, *cls_replace=None*, *use_xml_name=True*, *read_test=False*)

convert voc-like dataset to coco-like dataset

参数

- **root** (*str*) – path include images, xml, file list
- **mode** (*str*) – 'train', 'val', 'trainval', 'test'. used to find file list.
- **cls** (*str or list*) – class name in a file or a list.
- **cls_replace** (*dict*) – a dictionary for replacing class name. if not needed, you can just ignore it.
- **use_xml_name** (*bool*) – image filename source, if true, using the same name as xml for the image, otherwise using 'filename' in xml context for the image.
- **read_test** (*bool*) – Test if the picture can be read normally.

class cvtools.label_convert.DOTA2COCO(*label_root, image_root, classes=['large-vehicle', 'swimming-pool', 'helicopter', 'bridge', 'plane', 'ship', 'soccer-ball-field', 'basketball-court', 'ground-track-field', 'small-vehicle', 'harbor', 'baseball-diamond', 'tennis-court', 'roundabout', 'storage-tank'], path_replace=None, box_form='x1y1wh'*)

convert DOTA labels to coco-like format labels.

参数

- **label_root** (*str*) – label file path, for example, '/home/data/DOTA/train/labelTxt'
- **image_root** (*str*) – image path, for example, '/home/data/DOTA/train/images'
- **classes** (*str or list*) – class name in a file or a list.
- **path_replace** (*dict*) – replace same things in images path, if not needed, you can just ignore it.
- **box_form** (*str*) – coco bbox format, default 'x1y1wh'.

class cvtools.label_convert.COCO2Dets(*anns_file*, *num_coors=4*)

将 DOTA-COCO 兼容格式 GT 转成检测结果表达形式 results，保存成 pkl results: {

image_id: dets, # image_id 必须是 anns 中有效的 id image_id: dets, ...

} dets: {

cls_id：[[位置坐标，得分], [...], ...], cls_id: [[位置坐标，得分], [...], ...], ...

},

handle_ann(*ann*)

如果想自定义 ann 处理方式，继承此类，然后重新实现此方法

3.7.2 data_augs

class cvtools.data_augs.**Compose**(*transforms*)

Composes several augmentations together. :param transforms: list of transforms to compose. :type transforms: List[Transform]

Example

```
>>> augmentations.Compose([
>>>     transforms.CenterCrop(10),
>>>     transforms.ToTensor(),
>>> ])
```

class cvtools.data_augs.**RandomSampleCrop**

Crop :param img: the image being input during training :type img: Image :param boxes: the original bounding boxes in pt form :type boxes: Tensor :param labels: the class labels for each bbox :type labels: Tensor :param mode: the min and max jaccard overlaps :type mode: float tuple

返回

(img, boxes, classes) img (Image): the cropped image boxes (Tensor): the adjusted bounding boxes in pt form labels (Tensor): the class labels for each bbox

class cvtools.data_augs.**RandomRotate**

随机旋转 0 度、90 度、180 度、270 度

class cvtools.data_augs.**RandomVerMirror**

竖直方向（flipping around the x-axis）镜像

class cvtools.data_augs.**RandomHorMirror**

水平方向（flipping around the y-axis）镜像

3.7.3 label_analysis

class cvtools.label_analysis.**COCOAnalysis**(*img_prefix*, *ann_file=None*)

coco-like datasets analysis

vis_instances(*save_root*, *vis='bbox'*, *vis_cats=None*, *output_by_cat=False*, *box_format='x1y1wh'*)

Visualise bbox and polygon in annotation.

包含某一类的图片上所有类别均会被绘制。

参数

- **save_root** (*str*) – path for saving image.
- **vis** (*str*) – 'bbox' or 'segmentation'

- **vis_cats** (*list*) – categories to be visualized
- **output_by_cat** (*bool*) – output visual images by category.
- **box_format** (*str*) – 'x1y1wh' or 'polygon'

3.7.4 evaluation

cvtools.evaluation.**get_classes**(*dataset*)

Get class names of a dataset.

cvtools.evaluation.**average_precision**(*recalls*, *precisions*, *mode='area'*)

Calculate average precision (for single or multiple scales).

参数

- **recalls** (*ndarray*) – shape (num_scales, num_dets) or (num_dets,)
- **precisions** (*ndarray*) – shape (num_scales, num_dets) or (num_dets,)
- **mode** (*str*) – 'area' or '11points', 'area' means calculating the area under precision-recall curve, '11points' means calculating the average precision of recalls at [0, 0.1, ..., 1]

返回 calculated average precision

返回类型 float or ndarray

cvtools.evaluation.**eval_map**(*det_results*, *gt_bboxes*, *gt_labels*, *gt_ignore=None*, *scale_ranges=None*, *iou_thr=0.5*, *dataset=None*, *print_summary=True*, *calc_ious=<function bbox_overlaps>*)

Evaluate mAP of a dataset.

参数

- **det_results** (*list*) – a list of list, [[cls1_det, cls2_det, ...], ...] cls1_det 为 np.array，包含 K*5，包含得分, x1y1x2y2 形式
- **gt_bboxes** (*list*) – ground truth bboxes of each image, a list of K*4 array. x1y1x2y2 形式
- **gt_labels** (*list*) – ground truth labels of each image, a list of K array
- **gt_ignore** (*list*) – gt ignore indicators of each image, a list of K array
- **scale_ranges** (*list, optional*) – [(min1, max1), (min2, max2), ...]
- **iou_thr** (*float*) – IoU threshold，目前还不支持 polyiou
- **dataset** (*None or str or list*) – dataset name or dataset classes, there are minor differences in metrics for different datsets, e.g. "voc07", "imagenet_det", etc.
- **print_summary** (*bool*) – whether to print the mAP summary

返回 (mAP, [dict, dict, ...])

返回类型 tuple

cvtools.evaluation.**print_map_summary**(*mean_ap*, *results*, *dataset=None*)

Print mAP and results of each class.

参数

- **mean_ap** (*float*) – calculated from *eval_map*
- **results** (*list*) – calculated from *eval_map*
- **dataset** (*None or str or list*) – dataset name or dataset classes.

cvtools.evaluation.**eval_recalls**(*gts*, *proposals*, *proposal_nums=None*, *iou_thrs=None*, *print_summary=True*)

Calculate recalls.

参数

- **gts** (*list or ndarray*) – a list of arrays of shape (n, 4)
- **proposals** (*list or ndarray*) – a list of arrays of shape (k, 4) or (k, 5)
- **proposal_nums** (*int or list of int or ndarray*) – top N proposals
- **thrs** (*float or list or ndarray*) – iou thresholds

返回 recalls of different ious and proposal nums

返回类型 ndarray

cvtools.evaluation.**print_recall_summary**(*recalls*, *proposal_nums*, *iou_thrs*, *row_idxs=None*, *col_idxs=None*)

Print recalls in a table.

参数

- **recalls** (*ndarray*) – calculated from *bbox_recalls*
- **proposal_nums** (*ndarray or list*) – top N proposals
- **iou_thrs** (*ndarray or list*) – iou thresholds
- **row_idxs** (*ndarray*) – which rows(proposal nums) to print
- **col_idxs** (*ndarray*) – which cols(iou thresholds) to print

cvtools.evaluation.**plot_num_recall**(*recalls*, *proposal_nums*)

Plot Proposal_num-Recalls curve.

参数

- **recalls** (*ndarray or list*) – shape (k,)
- **proposal_nums** (*ndarray or list*) – same shape as *recalls*

cvtools.evaluation.**plot_iou_recall**(*recalls*, *iou_thrs*)

Plot IoU-Recalls curve.

参数

- **recalls** (*ndarray or list*) – shape (k,)
- **iou_thrs** (*ndarray or list*) – same shape as *recalls*

class cvtools.evaluation.**EvalCropQuality**(*ann_file*, *crop_ann_file*, *results=None*, *num_coors=4*)

此类设计目前不够完善，convert_crop_gt 应隐藏在内部

3.7.5 file_io

cvtools.file_io.**load_json**(*file*)

加载 json 文件

参数 file – 包含路径的文件名

Returns:

cvtools.file_io.**load_pkl**(*file*)

加载 pickle 序列化对象

参数 file – 包含路径的文件名

返回 unpickle object

Raises UnpicklingError

cvtools.file_io.**readlines**(*file*)

按行读取 str 到 list

参数 file – 包含路径的文件名

Returns:

cvtools.file_io.**read_file_to_list**(*file*)

读入单个文件输出 list，支持中文

参数 file – 包含路径的文件名

返回 所有文件内容放在 list 中返回

cvtools.file_io.**read_files_to_list**(*files*, *root=''*)

读入单个或多个文件合成一个 list 输出，支持中文

此函数设计是一个教训，只有必要的参数才能设计成位置参数，其它参数为关键字参数

参数

- **files** (*str*) – 文件名
- **root** (*root*) – 可选，文件名路径。如果指定 files 不可加路径

cvtools.file_io.**read_key_value**(*file*)

支持注释，支持中文

参数 **file** (*str*) – 包含路径的文件名

cvtools.file_io.**dump_json**(*data*, *to_file='data.json'*)

写 json 文件

参数

- **data** – 待保存成 json 格式的对象
- **to_file** – 保存的文件名

cvtools.file_io.**dump_pkl**(*data*, *to_file='data.pkl'*)

使用 pickle 序列化对象

参数

- **data** – 待序列化对象
- **to_file** – 保存的文件名

cvtools.file_io.**write_list_to_file**(*data*, *dst*, *line_break=True*)

保存 list 到文件

参数

- **data** (*list*) – list 中元素只能是基本类型
- **dst** (*str*) – 保存的文件名
- **line_break** – 是否加换行

Returns:

cvtools.file_io.**write_key_value**(*data*, *to_file*)

写字典到文件中（非序列化）

每行以字符':' 分割 key 和 value

参数

- **data** (*dict*) – dict 中元素只能是基本类型
- **to_file** – 保存的文件名

Returns:

cvtools.file_io.**write_str**(*data*, *to_file*)

写字符串到文件

参数

- **data** (*str*) – str 对象
- **to_file** (*str*) – 保存的文件名

3.7.6 utils

cvtools.utils.**get_files_list**(*root*, *file_type=None*, *basename=False*)

file_type is a str or list.

cvtools.utils.**makedirs**(*path*)

对 os.makedirs 进行扩展

从路径中创建文件夹，可创建多层。如果仅是文件名，则无须创建，返回 False；如果是已存在文件或路径，则无须创建，返回 False

参数 **path** – 路径，可包含文件名。纯路径最后一个字符需要是 os.sep

cvtools.utils.**find_in_path**(*name*, *path*)

Find a file in a search path

cvtools.utils.**imread**(*img_or_path*, *flag='color'*)

Read an image.

参数

- **img_or_path** (*ndarray or str*) – Either a numpy array or image path. If it is a numpy array (loaded image), then it will be returned as is.
- **flag** (*str*) – Flags specifying the color type of a loaded image, candidates are *color*, *grayscale* and *unchanged*.

返回 Loaded image array.

返回类型 ndarray

cvtools.utils.**imwrite**(*img*, *file_path*, *params=None*, *auto_mkdir=True*)

Write image to file

参数

- **img** (*ndarray*) – Image array to be written.
- **file_path** (*str*) – Image file path.
- **params** (*None or list*) – Same as opencv's `imwrite()` interface.
- **auto_mkdir** (*bool*) – If the parent folder of *file_path* does not exist, whether to create it automatically.

返回 Successful or not.

返回类型 bool

cvtools.utils.**draw_boxes_texts**(*img*, *boxes*, *texts=None*, *colors=None*, *line_width=1*, *draw_start=False*, *box_format='x1y1x2y2'*)

Draw bboxes on an image.

参数

- **img** (*str or ndarray*) – The image to be displayed.
- **boxes** (*list or ndarray*) – A list of ndarray of shape (k, 4).
- **texts** (*list*) – A list of shape (k).
- **colors** (*list[tuple or Color]*) – A list of colors.
- **line_width** (*int*) – Thickness of lines.
- **draw_start** (*bool*) – Draw a dot at the first vertex of the box.
- **box_format** (*str*) – x1y1x2y2(default), x1y1wh, xywh, xywha, polygon

cvtools.utils.**draw_class_distribution**(*y*, *save_name='class_distribution.png'*)

绘制饼图, 其中 y 是标签列表

cvtools.utils.**draw_hist**(*data*, *bins=10*, *x_label='区间'*, *y_label='频数/频率'*, *title='频数/频率分布直方图'*, *show=True*, *save_name='hist.png'*, *density=True*)

绘制直方图 data: 必选参数，绘图数据 bins: 直方图的长条形数目，可选项，默认为 10

cvtools.utils.**x1y1wh_to_x1y1x2y2**(*xywh*)

Convert [x1 y1 w h] box format to [x1 y1 x2 y2] format. supported type: list, type and np.ndarray

cvtools.utils.**x1y1x2y2_to_x1y1wh**(*xyxy*)

Convert [x1 y1 x2 y2] box format to [x1 y1 w h] format.

cvtools.utils.**xywh_to_x1y1x2y2**(*xywh*)

Convert [x y w h] box format to [x1 y1 x2 y2] format.

cvtools.utils.**x1y1x2y2_to_xywh**(*x1y1x2y2*)

Convert [x1 y1 x2 y2] box format to [x y w h] format.

cvtools.utils.**x1y1wh_to_xywh**(*x1y1wh*)

Convert [x1 y1 w h] box format to [x y w h] format. supported type: list, type and np.ndarray

cvtools.utils.**rotate_rect**(*rect*, *center*, *angle*)

一个数学问题：2x2 矩阵（坐标）与旋转矩阵相乘. 在笛卡尔坐标系中，angle>0, 逆时针旋转; angle<0, 顺时针旋转

参数

- **rect** – x1y1x2y2 形式矩形
- **center** – 旋转中心点
- **angle** – 旋转角度，范围在 (-180, 180)

返回 x1y1x2y2x3y3x4y4 format box

cvtools.utils.**xywha_to_x1y1x2y2x3y3x4y4**(*xywha*)

用旋转的思路做变换是最通用和最简单的

警告：目前多维一起操作还有些问题!

参数 **xywha** – (5,) 一维 list 或 (K, 5) 多维 array

class cvtools.utils.Timer

A simple timer.

cvtools.utils.get_time_str(*form='%Y%m%d_%H%M%S'*)

for example form='%Y%m%d_%H%M%S_%f'

cvtools.utils.bbox_overlaps(*bboxes1*, *bboxes2*, *mode='iou'*)

Calculate the ious between each bbox of bboxes1 and bboxes2.

参数

- **bboxes1** (*ndarray*) – shape (n, 4)
- **bboxes2** (*ndarray*) – shape (k, 4)
- **mode** (*str*) – iou (intersection over union) or iof (intersection over foreground)

返回 shape (n, k)

返回类型 ious(ndarray)

cvtools.utils.is_str(*x*)

Whether the input is an string instance.

cvtools.utils.iter_cast(*inputs*, *dst_type*, *return_type=None*)

Cast elements of an iterable object into some type.

参数

- **inputs** (*Iterable*) – The input object.
- **dst_type** (*type*) – Destination type.
- **return_type** (*type, optional*) – If specified, the output object will be converted to this type, otherwise an iterator.

返回 The converted object.

返回类型 iterator or specified type

cvtools.utils.list_cast(*inputs*, *dst_type*)

Cast elements of an iterable object into a list of some type.

A partial method of `iter_cast()`.

cvtools.utils.tuple_cast(*inputs*, *dst_type*)

Cast elements of an iterable object into a tuple of some type.

A partial method of `iter_cast()`.

cvtools.utils.is_seq_of(*seq*, *expected_type*, *seq_type=None*)

Check whether it is a sequence of some type.

参数

- **seq** (*Sequence*) – The sequence to be checked.
- **expected_type** (*type*) – Expected type of sequence items.
- **seq_type** (*type, optional*) – Expected sequence type.

返回 Whether the sequence is valid.

返回类型 bool

cvtools.utils.**is_list_of**(*seq*, *expected_type*)

Check whether it is a list of some type.

A partial method of `is_seq_of()`.

cvtools.utils.**is_tuple_of**(*seq*, *expected_type*)

Check whether it is a tuple of some type.

A partial method of `is_seq_of()`.

cvtools.utils.**slice_list**(*in_list*, *lens*)

Slice a list into several sub lists by a list of given length.

参数

- **in_list** (*list*) – The list to be sliced.
- **lens** (*int or list*) – The expected length of each out list.

返回 A list of sliced list.

返回类型 list

cvtools.utils.**concat_list**(*in_list*)

Concatenate a list of list into a single list.

参数 **in_list** (*list*) – The list of list to be merged.

返回 The concatenated flat list.

返回类型 list

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

待处理: 测试

(原始记录 见 /home/docs/checkouts/readthedocs.org/user_builds/cvtools/checkouts/stable/docs/utils/file.rst, 第 5 行。)

待处理: 字符串文档更新

(原始记录 见 /home/docs/checkouts/readthedocs.org/user_builds/cvtools/checkouts/stable/docs/utils/file.rst, 第 7 行。)

待处理: 文档更新

(原始记录 见 /home/docs/checkouts/readthedocs.org/user_builds/cvtools/checkouts/stable/docs/utils/file.rst, 第 9 行。)

索引

P

R

S

T

V

W

X