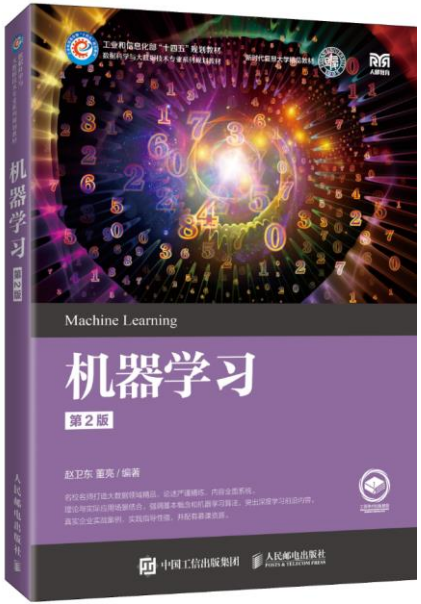
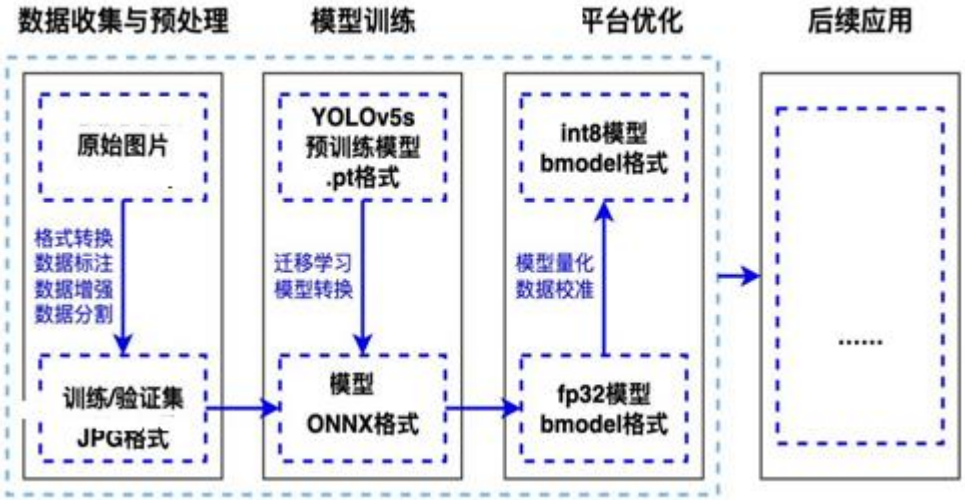


目标检测应用的基本过程

赵卫东 复旦大学

整体思路

- 目标检测的整体过程分为数据收集与预处理、模型训练、算能平台优化等环节。



业务背景分析

➢需要明确本项目要解决的问题，这是数据选择和算法选择的依据。

数据集的准备

- 预训练的目标检测模型可能不能达到具体应用的性能要求。
- 需要补充与问题相关的数据，可以通过拍摄、爬取等手段手机，并要人工标注。

利用labelImg进行数据标注

- 在训练深度学习模型时，需要大量带有标注的样本。LabelImg是一款便捷的标注工具，需要借此工具标注图像中的目标位置和类别。
- 标注数据会以json文件的形式进行保存，记录了每个检测框对应的物体类别、检测框的中心点坐标和长、宽属性。

利用labelImg进行数据标注（1）

github.com/tzutalin/labelImg

Search or jump to... / Pull requests Issues Marketplace Explore

tzutalin / labelImg Public

<> Code Issues 316 Pull requests 36 Actions Projects Wiki Security Insights

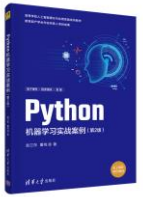
master 5 branches 25 tags Go to file Add file Code

zhangchn UX: Increase size of vertex controls around shape (#863) ... 861fbe9 on 30 Apr 429 commits

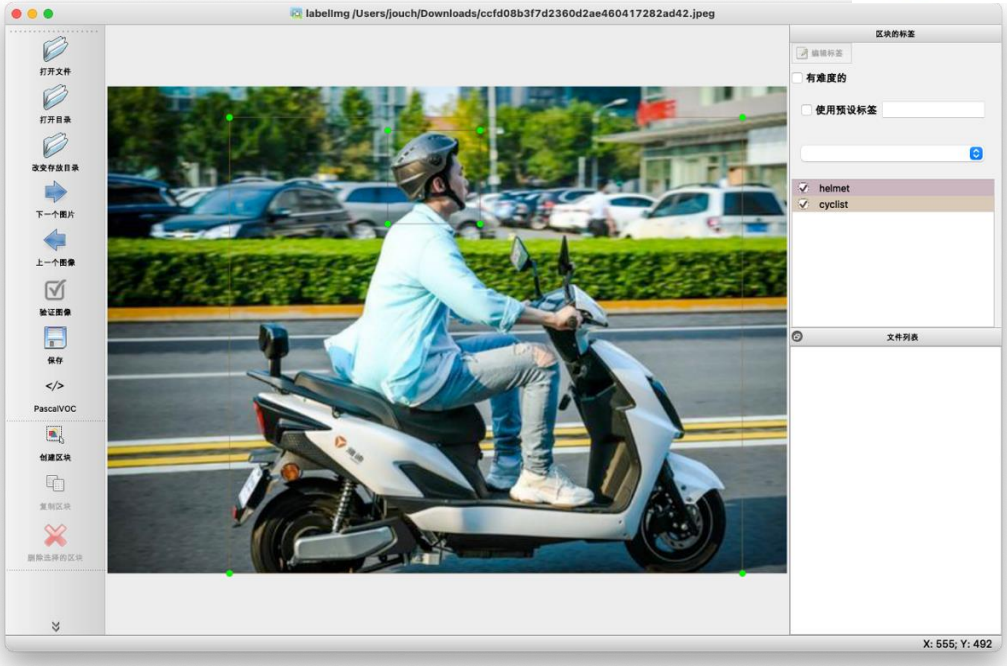
.github	Update no-response.yml	12 months ago
build-tools	Fix typo	3 years ago
data	Modified the default label text box into a drop down (#824)	4 months ago
demo	Screenshot of macOS High Sierra usage added.	5 years ago
libs	UX: Increase size of vertex controls around shape (#863)	last month
readme	update instruction to install with pipenv	8 months ago
requirements	Bump lxml from 4.6.3 to 4.6.5 in /requirements (#830)	5 months ago
resources	update translations for zh-CN	3 months ago
tests	Update unit test to fix error and fix icon issue	14 months ago
tools	move convert dir to tools dir	16 months ago
.gitignore	Update GitIgnore / .DS_Store Removal	2 years ago
.travis.yml	Travis CI: Trusty, Python 3.5, and sudo on Travis are all end of life (...	15 months ago
CONTRIBUTING.rst	Create new files for pypackage	5 years ago
HISTORY.rst	bump version to 1.8.6	8 months ago
LICENSE	Create new files for pypackage	5 years ago
MANIFEST.in	Add missing files	5 years ago
Makefile	Update README.md and setup.py	3 years ago
README.rst	Problems found while packaging as binaries.Has been tested. (#855)	3 months ago
__init__.py	Create new files for pypackage	5 years ago
issue_template.md	Create issue_template.md	5 years ago

https://github.com/tzutalin/labelImg

利用labelImg进行数据标注（2）

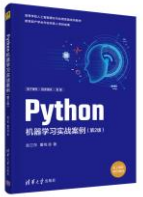


- 在训练深度学习模型时，需要大量带有标注的样本。LabelImg是一款便捷的标注工具，需要借此工具标注图像中的目标位置和类别。
- labelImg工具数据集的标注格式为Pascal VOC，标注会生成XML标签文件。



精灵标注助手

利用labelImg进行数据标注（3）




```
<annotation>
    <folder>Downloads</folder>
    <filename>ccfd08b3f7d2360d2ae460417282ad42.jpeg</filename>
    <path>/Users/jouch/Downloads/ccfd08b3f7d2360d2ae460417282ad42.jpeg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>640</width>
        <height>453</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>helmet</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>260</xmin>
            <ymin>36</ymin>
            <xmax>354</xmax>
            <ymax>127</ymax>
        </bndbox>
    </object>
    <object>
        <name>cyclist</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>107</xmin>
            <ymin>24</ymin>
            <xmax>609</xmax>
            <ymax>452</ymax>
        </bndbox>
    </object>
</annotation>
```

数据集格式转换

- 由labelImg软件生成的json格式并不能直接用于YOLOv5的训练，还需要将其转为YOLO格式的txt文件。

```
classes = ["感冒灵胶囊", "头孢克肟胶囊", "布洛芬缓释胶囊", "连花清瘟胶囊", "复方氨酚烷胺片", "头孢地尼胶囊", "京都念慈庵金桔柠檬糖", "未正面放置"]

def json2yolo(filenum, label, width, height):
    f = open("dataset_origin/augmentation/label/" + str(filenum).zfill(5) + ".txt", 'w')
    for annotation in label[0]['annotations']:
        x = annotation["coordinates"]["x"] / width
        y = annotation["coordinates"]["y"] / height
        w = annotation["coordinates"]["width"] / width
        h = annotation["coordinates"]["height"] / height
        class_num = classes.index(annotation["label"])
        str_box = str(class_num)+' '+format(x, '.6f')+' '+format(y, '.6f')+' '+format(w, '.6f')+' '+format(h, '.6f')+'\n'
        f.write(str_box)
    f.close()
```

```
import json
obj = json.load(open("dataset_origin/JPG/401672196279_.pic.json", 'r', encoding='utf-8'))
obj[0]['annotations']
```

[8]

```
[{'label': '感冒灵胶囊',
  'coordinates': {'x': 528.5,
   'y': 348.82352941176475,
   'width': 694.0,
   'height': 492.0}},
 {'label': '头孢克肟胶囊',
  'coordinates': {'x': 1170.0,
   'y': 335.32352941176475,
   'width': 653.0,
   'height': 647.0}},
 {'label': '布洛芬缓释胶囊',
  'coordinates': {'x': 499.0,
   'y': 910.8235294117649,
   'width': 747.0,
   'height': 674.0000000000001}},
 {'label': '连花清瘟胶囊',
  'coordinates': {'x': 1224.0,
   'y': 920.8235294117649,
   'width': 797.0,
   'height': 700.0000000000001}}]
```

YOLO数据格式

➢准备labels，把数据集格式转换成YOLO_txt格式，即将每个xml标注提取bbox信息为txt格式，每个图像对应一个txt文件，文件每一行为一个目标的信息，包括类别 xmin xmax ymin ymax。

001.txt — 已编辑

```
0 0.49895833333333334 0.5328125 0.9979166666666667 0.203125
```

数据增强的必要性

- 为什么要补充数据对预训练的目标检测模型进行训练？
- 机器学习的本质
- 提升模型的鲁棒性

数据增强

- 数据增强方式包括常用的随机亮度、随机对比度、随机饱和度、随机色调、随机翻转、随机旋转、Mosaic等方法。


```python
def random_rotate90(img, label):
    rotate90 = random.uniform(0,1) >= 0.5
    rotate270 = random.uniform(0,1) >= 0.5

    width, height = img.size
    if rotate90:
        # 顺时针旋转90度
        img = img.transpose(Image.ROTATE_270)
        for annotation in label[0]['annotations']:
            org_x = annotation["coordinates"]["x"]
            org_y = annotation["coordinates"]["y"]
            org_w = annotation["coordinates"]["width"]
            org_h = annotation["coordinates"]["height"]
            annotation["coordinates"]["x"] = height - org_y
            annotation["coordinates"]["y"] = org_x
            annotation["coordinates"]["width"] = org_h
            annotation["coordinates"]["height"] = org_w


    elif rotate270:
        # 逆时针旋转90度
        img = img.transpose(Image.ROTATE_90)
        for annotation in label[0]['annotations']:
            org_x = annotation["coordinates"]["x"]
            org_y = annotation["coordinates"]["y"]
            org_w = annotation["coordinates"]["width"]
            org_h = annotation["coordinates"]["height"]
            annotation["coordinates"]["x"] = org_y
            annotation["coordinates"]["y"] = width - org_x
            annotation["coordinates"]["width"] = org_h
            annotation["coordinates"]["height"] = org_w
    return img, label
```

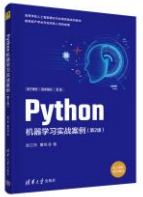
```python
def random_filp(img, label):
    horizon_flip = random.uniform(0,1) >= 0.5
    vertical_flip = random.uniform(0,1) >= 0.5

    width, height = img.size
    if horizon_flip:
        img = img.transpose(Image.FLIP_LEFT_RIGHT)
        for annotation in label[0]['annotations']:
            annotation["coordinates"]["x"] = width - annotation["coordinates"]["x"]

    if vertical_flip:
        img = img.transpose(Image.FLIP_TOP_BOTTOM)
        for annotation in label[0]['annotations']:
            annotation["coordinates"]["y"] = height - annotation["coordinates"]["y"]
    return img, label
```

数据集分割

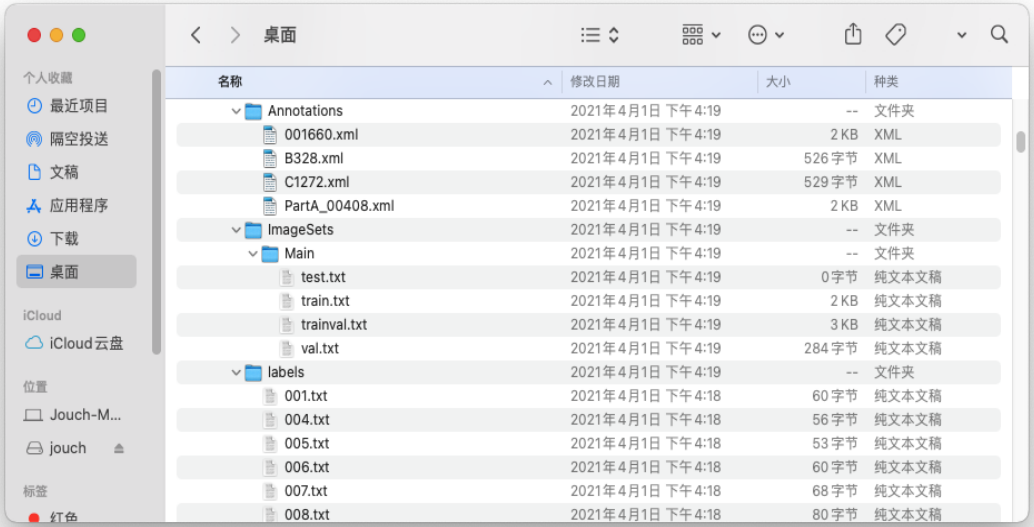
- 将原始的图片集按照一定的比例划分为训练集（或者校验集）和检测集，然后随机打乱，得到训练集和检测集。

```
import random
import shutil

img_list = os.listdir("dataset_origin/augmentation/image")
img_list.sort(key = lambda x: int(x[:-4]))

random.shuffle(img_list)
for file in img_list[:2500]:
    filename = file.split('.')[0]
    img_src = "dataset_origin/augmentation/image/"+file
    img_tgt = "yolov5/data/mydataset/train/images/"+file
    shutil.copyfile(img_src, img_tgt)
    label_src = "dataset_origin/augmentation/label/"+filename+".txt"
    label_tgt = "yolov5/data/mydataset/train/labels/"+filename+".txt"
    shutil.copyfile(label_src, label_tgt)
for file in img_list[2500:]:
    filename = file.split('.')[0]
    img_src = "dataset_origin/augmentation/image/"+file
    img_tgt = "yolov5/data/mydataset/val/images/"+file
    shutil.copyfile(img_src, img_tgt)
    label_src = "dataset_origin/augmentation/label/"+filename+".txt"
    label_tgt = "yolov5/data/mydataset/val/labels/"+filename+".txt"
    shutil.copyfile(label_src, label_tgt)
# os.listdir("dataset_origin/augmentation/label")
```

数据划分



➢创建存储训练数据的文件夹，文件夹内的目录结构如下，将之前labelImg标注好的xml文件和图片放到对应目录下。



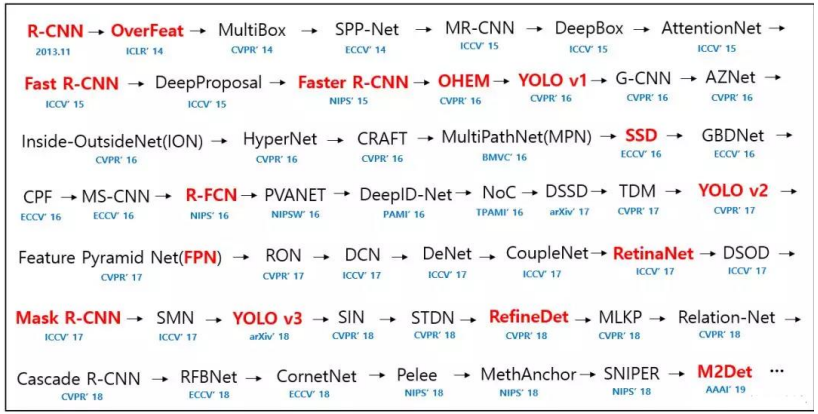
数据集配置

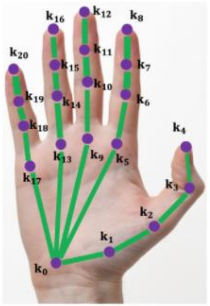
- 对数据集进行配置，在YOLO v5目录下的data文件夹下新建一个ab.yaml文件，用来存放训练集和验证集的划分文件（train.txt和val.txt），还有目标的类别数目和具体类别列表。

```
train: /home/nankai02/YOLO v5-4.0/paper  data/train.txt
val: /home/nankai02/YOLO v5-4.0/paper  data/val.txt
nc: 2
names: ['crack','helmet']
```

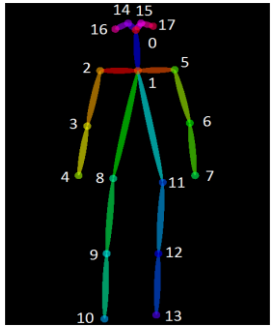
目标检测算法选择

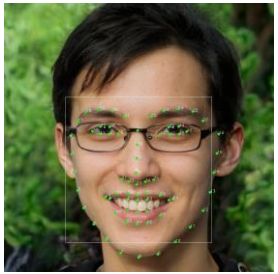
姿态估计、目标检测、目标追踪、关键点检测

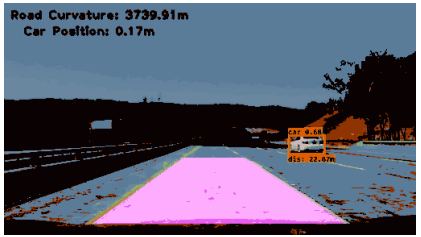




人脸特征关键点识别

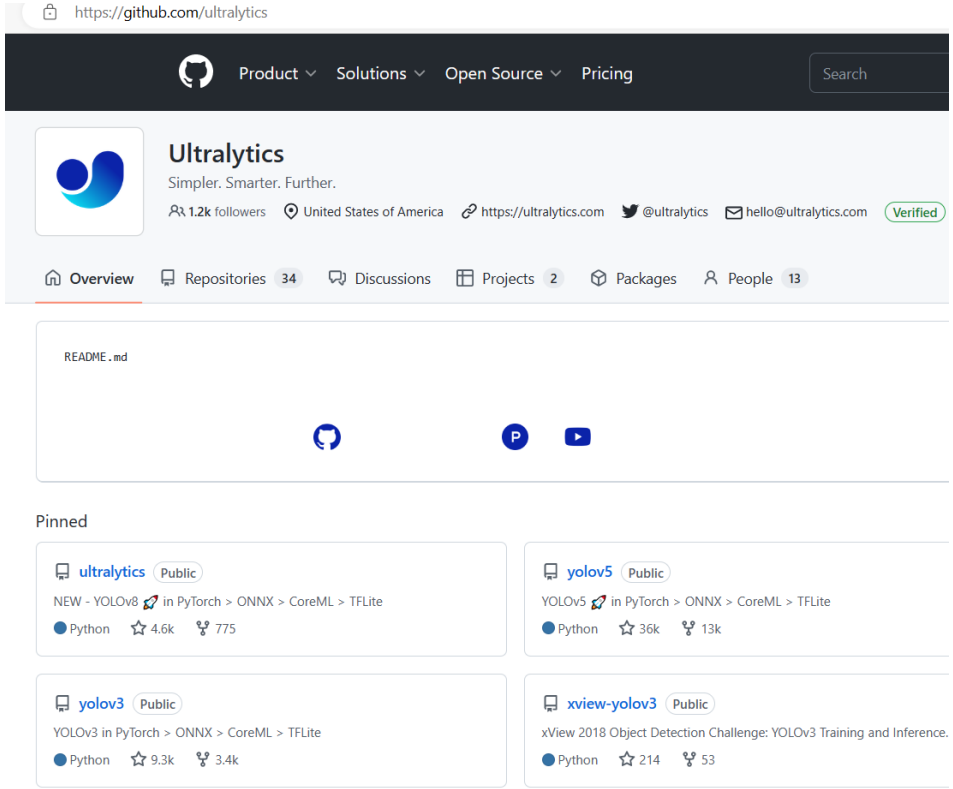






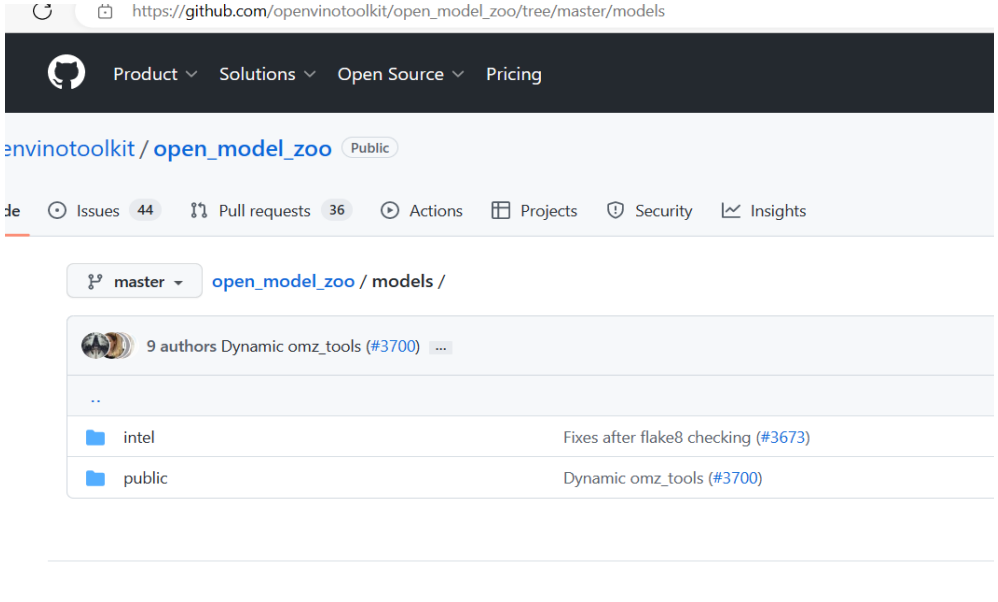
模型选择

- 通过比较，选择了目前较成熟的YOLOv5算法。YOLOv5模型保持了和YOLOv4相当的检测性能的同时，在模型的体积和灵活性上都进行了优化，
- 在YOLOv5的五种版本中，本案例选择了YOLOv5s模型，在检测性能、体积和速度等指标上都达到了较好的折中。



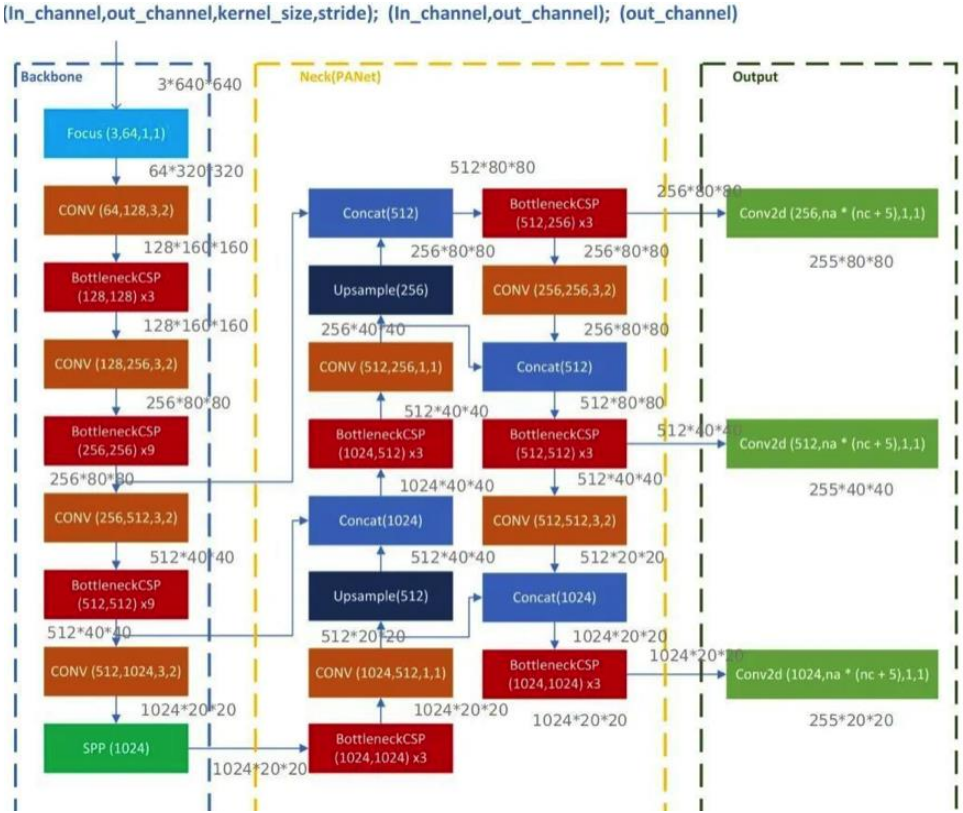
模型下载地址

- 预训练模型下载地址：
- https://github.com/ultralytics/yolov5
- https://github.com/openvinotoolkit/open_model_zoo/tree/master/models



YOLO v5模型结构

- **Backbone**：在不同图像特征粒度上卷积，并形成图像特征的卷积神经网络。
- **Neck**：一系列组合图像特征的网络层，并将图像特征传递到预测层 (一般是FPN或者PANET) 。
- **Head**：对图像特征进行预测，生成边界框并预测类别。



YOLO v5模型结构

YOLO v5使用了多种数据预处理方法：

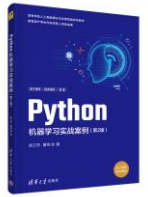
（1）Mosaic数据增强：采用随机缩放、随机裁剪、随机排布的方式进行拼接。

（2）自适应anchor：在YOLOv3、YOLOv4中，训练不同的数据集时，计算初始锚框的值是通过单独的程序运行的。但YOLO v5中将此功能嵌入到代码中，每次训练时，自适应地计算不同训练集中的最佳锚框值。

（3）自适应图片：在常用的目标检测算法中，不同的图片长宽都不相同，因此常用的方式是将原始图片统一缩放到一个标准尺寸，再送入检测网络中。但YOLO v5代码中对此进行了改进，也是YOLO v5推理速度能够很快的一个不错的trick。

（4）正样本增加：增加高质量正样本anchor，显著加速收敛。

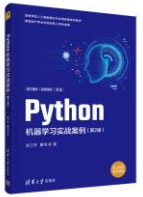
YOLO v5模型训练（1）



➢调整配置文件的anchor和类别数

```yaml
# parameters
nc: 2  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32
```

YOLO v5模型训练（2）



- 源码中YOLO v5目录下的weights文件夹提供了下载四种预训练模型的脚本download_weights.sh，执行这个shell脚本就可以下载。对train.py文件进行修改。

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default='yolov5_weights/yolov5s.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='models/yolov5s.yaml', help='model.yaml path')
    parser.add_argument('--data', type=str, default='data/ab.yaml', help='data.yaml path')
    parser.add_argument('--hyp', type=str, default='data/hyp.scratch.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=300)
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs')
    parser.add_argument('--img-size', nargs='+', type=int, default=[640, 640], help='[train, test] image sizes')
    parser.add_argument('--rect', action='store_true', help='rectangular training')
    parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')
    parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
    parser.add_argument('--notest', action='store_true', help='only test final epoch')
    parser.add_argument('--noautoanchor', action='store_true', help='disable autoanchor check')
    parser.add_argument('--evolve', action='store_true', help='evolve hyperparameters')
    parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
    parser.add_argument('--cache-images', action='store_true', help='cache images for faster training')
    parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
    parser.add_argument('--name', default='', help='renames results.txt to results_name.txt if supplied')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')
    parser.add_argument('--single-cls', action='store_true', help='train as single-class dataset')
    parser.add_argument('--adam', action='store_true', help='use torch.optim.Adam() optimizer')        # action='sto
    parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
    parser.add_argument('--local_rank', type=int, default=-1, help='DDP parameter, do not modify')
    parser.add_argument('--logdir', type=str, default='runs/', help='logging directory')
    parser.add_argument('--workers', type=int, default=8, help='maximum number of dataloader workers')
```

模型训练

- 本案例采用迁移学习的方式，对YOLOv5s预训练模型在上述数据集上进行了训练。迁移学习分为两个阶段：
  - 在第一个阶段，预训练YOLOv5s模型的backbone网络参数被冻结，只对最后几层分类与检测网络的参数进行训练；
  - 在第二个阶段，模型的所有参数解冻，再重新在数据集上进行参数的微调。

```
python train.py --data coco.yaml --cfg yolov5n.yaml --weights '' --batch-size 128
                                       yolov5s                                    64
                                       yolov5m                                    40
                                       yolov5l                                    24
                                       yolov5x                                    16
```

```python
# Freeze
freeze = [f'model.{x}.' for x in (freeze if len(freeze) > 1 else range(freeze[0]))]  # layers to freeze
for k, v in model.named_parameters():
    v.requires_grad = True  # train all layers
    if any(x in k for x in freeze):
        LOGGER.info(f'freezing {k}')
        v.requires_grad = False
```

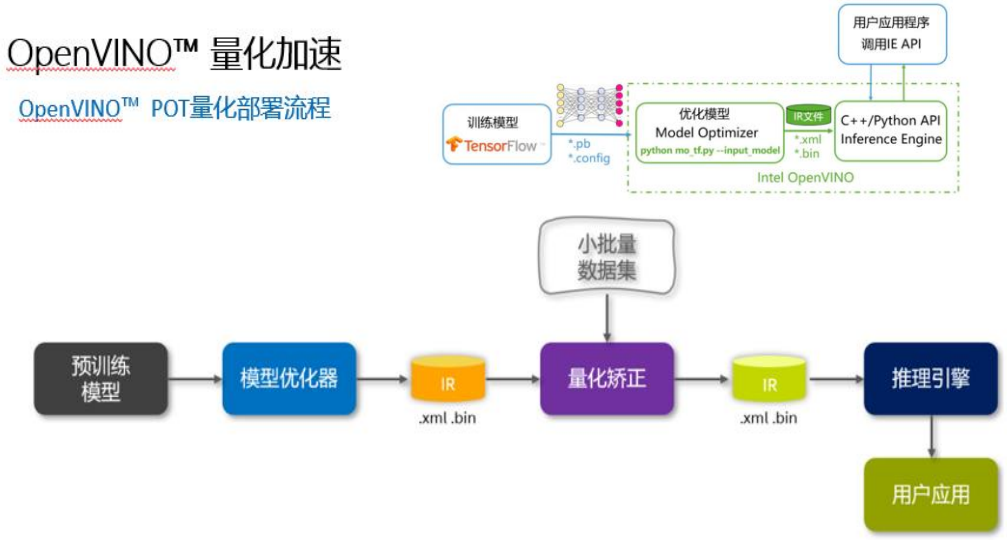
YOLOv5模型应用

➢通过YOLO v5下的detect.py加载模型,进行图片的测试。

```
python detect.py --weights yolov5s.pt                 # PyTorch
                           yolov5s.torchscript        # TorchScript
                           yolov5s.onnx               # ONNX Runtime or OpenCV DNN with --dnn
                           yolov5s_openvino_model     # OpenVINO
                           yolov5s.engine             # TensorRT
                           yolov5s.mlmodel            # CoreML (macOS only)
                           yolov5s_saved_model        # TensorFlow SavedModel
                           yolov5s.pb                 # TensorFlow GraphDef
                           yolov5s.tflite             # TensorFlow Lite
                           yolov5s_edgetpu.tflite     # TensorFlow Edge TPU
                           yolov5s_paddle_model       # PaddlePaddle
```

模型转换

- 在将模型部署到算能平台进行优化之前，本案例需要将模型先转换为ONNX格式。经过ONNX转换出来的算子可以在算能平台的TPU硬件上完成加速。



OpenVINO下载地址https://www.intel.cn/content/www/cn/zh/developer/tools/openvino-toolkit/overview.html

模型移植

➢ **pt->onnx**

利用YOLO v5的models文件夹下的export.py进行模型的转换，将pt模型转化成onnx，onnx是开放神经网络交换（open neural network exchange）格式，是一个用于表示深度学习模型的标准，可以使模型在不同框架之间进行转化，将模型转化成万能格式。

安装onnx

Pip install onnx

```
python export.py --weights yolov5s.pt
```

fp32模型转换（1）

- 模型在算能平台完成部署需要将模型转换为fp32格式的.bmodel文件，即模型的各个参数还是以32位浮点数的形式存储在文件中。
- 通过bmneto工具指定ONNX模型路径以及对应运行模型推理的芯片型号，即可输出fp32格式的.bmodel模型文件。

fp32模型转换（2）

- 通过bm_model.bin --info {bmodel路径}命令，就可以查看转换出来的模型属性。

```
t/ultralytics_yolov5/runs/train/zyxu/compilation.bmodel
bmodel version: B.2.2
chip: BM1684
create time: Fri Dec 30 17:14:21 2022

=========================================
net 0: [best.onnx]  static
------------
stage 0:
input: images, [1, 3, 640, 640], float32, scale: 1
output: output0, [1, 25200, 13], float32, scale: 1

device mem size: 57885824 (coeff: 28759424, instruct: 216832, runtime: 28909568)
host mem size: 0 (coeff: 0, runtime: 0)
```

int8模型量化（1）

- 在得到fp32模型之后，算能平台能够基于这个模型进一步将其量化成int8模型，也就是模型中的参数都用8位定点数来存储。
- 这部分量化使用的是训练后量化的技术，需要一定数量的图片来完成校准。本案例从数据集中分出了50张图片作为校准图片，使用python3 -m ufw.cali.cali_model --model {ONNX文件路径} --cali_image_path {校准集路径} 命令，将fp32模型转换成了int8模型。

int8模型量化（2）

- 通过bm_model.bin --info {bmodel路径}命令，就可以查看转换出来的int8模型属性。

  int8模型的device mem size大约是fp32模型的一半。

- 在一张图片上运行对int8模型的推理，可以看到int8模型的推理时间大约为0.02秒，折算成帧率约为50fps，在fp32模型的基础上又有了接近一倍的提升，并且已经超出对模型实时性的基本要求

```
t/ultralytics_yolov5/runs/train/zyxu/yolov5s/compilation.bmodel
bmodel version: B.2.2
chip: BM1684
create time: Fri Dec 30 19:11:03 2022

=========================================
net 0: [yolov5s]  static
------------
stage 0:
input: images, [1, 3, 640, 640], int8, scale: 0.499625
output: output0, [1, 25200, 13], float32, scale: 1

device mem size: 36181760 (coeff: 7922688, instruct: 197376, runtime: 28061696)
```

```
02814.jpg
img pre cost time 0.028455495834350586
use decode data as input
input_data shape:  (1, 3, 640, 640)
(1, 25200, 13)
net cost time 0.021338462829589844
post cost time 0.0002963542938232422
(1707, 1280, 3)
==================================================
[2022-12-30 19:48:03.710] [info] [tensor.cpp:96] Start delete_shaptr_bm_handle_t
!
[2022-12-30 19:48:03.711] [info] [tensor.cpp:98] End delete_shaptr_bm_handle_t!
[2022-12-30 19:48:03.711] [info] [tensor.cpp:102] Start delete_shaptr_bm_handle_
t_allocated!
[2022-12-30 19:48:03.711] [info] [tensor.cpp:105] End delete_shaptr_bm_handle_t_
allocated!
```