**PyTorch-YOLOv3-tiny框架讲解**

Version PyTorch-YOLOv3-tiny By dreamofTaotao in 2019-09-18

**目录**

[PyTorch-YOLOv3框架目录注解：](#_Toc26812_WPSOffice_Level1) [2](#_Toc26812_WPSOffice_Level1)

[PyTorch-YOLOv3主目录README.md翻译+详解：](#_Toc29190_WPSOffice_Level1) [3](#_Toc29190_WPSOffice_Level1)

[附录一：](#_Toc22235_WPSOffice_Level1) [8](#_Toc22235_WPSOffice_Level1)

**PyTorch-YOLOv3框架目录注解：**

* assets:目标检测输出根目录
* config
  + coco.data:coco数据
  + create\_custom\_model.sh:自定义模型下载文件
  + custom.data:自定义数据
  + yolov3-tint.cfg:小版 yolov3配置文件
  + yolov3.cfg:yolov3配置文件
* data
  + custom: (自定义数据集)
    - image(数据集根目录)
      * train.jpg:数据
    - labels:(数据标注集)
      * train.txt:数据标注文件
    - classes.names:类名文件
    - train.txt:训练集目录文件
    - valid.txt:验证集目录文件
  + samples: (简单数据集，一些例子)
  + coco.names: coco数据集类名
  + get\_coco\_datasets.sh:下载coco数据集
* utils:(工具类，dataset.py文件将image图片转换为RGB通道)
  + \_\_init\_\_.py:初始化coco数据集
  + augmentations:转换函数
  + datasets.py:将image文件转换成RGB通道
  + logger.py:回调50和75。产生更好的效果
  + parse\_config.py:获取配置文件相关配置信息和数据集路径信息
  + utils.py:工具类
* weights:(初始化权重文件)
  + download\_weights.sh:下载初始化权重文件
* detect.py:检测文件并且保存相应的图片文件名
* models.py:模型选择文件，根据相应传入的参数
* requirements.txt:需求文件
* test.py:验证启动文件
* train.py:训练启动文件

**PyTorch-YOLOv3主目录README.md翻译+详解：**

**PyTorch-YOLOv3：**

PyTorch-YOLOv3是一个YOLOv3小型PyTorch接口，支持训练、预测和评估。

**Installation：(安装)**

1.克隆PyThor-YOLOv3项目并且安装项目中相应依赖的库

# 克隆本项目到本地

$ git clone https://github.com/eriklindernoren/PyTorch-YOLOv3

# 进入PyTorch-YOLOv3目录下

$ cd PyTorch-YOLOv3/

# 安装相应需求文件，也就是Python中的一些库文件，详细请看附录一

$ sudo pip3 install -r requirements.txt

2.下载预训练权重模型

# 进入weights文件目录下

$ cd weights/

# 下载预训练权重模型，直接使用Git Bash Here就可以下载

$ bash download\_weights.sh

3.下载COCO数据集

# 进入data目录下

$ cd data/

# 下载coco数据集，直接使用Git Bash Here就可以下载

$ bash get\_coco\_dataset.sh

**Test：(测试)**

在COCO测试中预估当前的模型。

# 使用test.py测试文件测试权重，weights\_path：预训练权重模型路径

$ python3 test.py --weights\_path weights/yolov3.weights

**Inference：(预测)**

使用当前的预训练权重模型预测图片。下面的表格展示了当输入图片达到256\*256的时候所要花费的时间。ResNet检测数据的来自于YOLOv3论文中。Darknet-53模型的测量使用的是原文作者的1080tiGPU加速器检测所花费的时间。

|  |  |  |
| --- | --- | --- |
| Backbone | GPU | FPS |
| ResNet-101 | Titan X | 53 |
| ResNet-152 | Titan X | 37 |
| Darknet-53(paper)  【原论文】 | Titan X | 76 |
| Darknet-53(this impl.)  【该项目】 | 1080ti | 74 |

# 使用detect.py检测目标数据集，image\_folder：检测目标图片路径

$ python3 detect.py --image\_folder data/samples/

**Train：(训练)**

# 训练模型

$ train.py [-h] [--epochs EPOCHS] [--batch\_size BATCH\_SIZE]

[--gradient\_accumulations GRADIENT\_ACCUMULATIONS]

[--model\_def MODEL\_DEF] [--data\_config DATA\_CONFIG]

[--pretrained\_weights PRETRAINED\_WEIGHTS] [--n\_cpu N\_CPU]

[--img\_size IMG\_SIZE]

[--checkpoint\_interval CHECKPOINT\_INTERVAL]

[--evaluation\_interval EVALUATION\_INTERVAL]

[--compute\_map COMPUTE\_MAP]

[--multiscale\_training MULTISCALE\_TRAINING]

# epoch:迭代次数

# batch\_size批量处理数据的大小

# gradient\_accumulations:导数累加

# model\_def:使用的模型

# data\_config:数据集配置

# pretrained\_weights:预训练权重

# --n\_cpu:加速器选择

# img\_size:图像大小

# checkpoint\_internal:检验间隔时长

# evaluation\_internal:验证间隔时长

# compute\_map:计算图显示

# multiscale\_training: 多尺度训练

Example(COCO)

使用Darknet-53预训练图像网络，运行代码：

# data\_config:数据集设置;pretrained\_weights:预训练权重模型

$ python3 train.py --data\_config config/coco.data --pretrained\_weights weights/darknet53.conv.74

Training log(打印训练日志)

---- [Epoch 7/100, Batch 7300/14658] ----

+------------+--------------+--------------+--------------+

| Metrics | YOLO Layer 0 | YOLO Layer 1 | YOLO Layer 2 |

+------------+--------------+--------------+--------------+

| grid\_size | 16 | 32 | 64 |

| loss | 1.554926 | 1.446884 | 1.427585 |

| x | 0.028157 | 0.044483 | 0.051159 |

| y | 0.040524 | 0.035687 | 0.046307 |

| w | 0.078980 | 0.066310 | 0.027984 |

| h | 0.133414 | 0.094540 | 0.037121 |

| conf | 1.234448 | 1.165665 | 1.223495 |

| cls | 0.039402 | 0.040198 | 0.041520 |

| cls\_acc | 44.44% | 43.59% | 32.50% |

| recall50 | 0.361111 | 0.384615 | 0.300000 |

| recall75 | 0.222222 | 0.282051 | 0.300000 |

| precision | 0.520000 | 0.300000 | 0.070175 |

| conf\_obj | 0.599058 | 0.622685 | 0.651472 |

| conf\_noobj | 0.003778 | 0.004039 | 0.004044 |

+------------+--------------+--------------+--------------+

Total Loss 4.429395

---- ETA 0:35:48.821929

Tensorboard

跟踪训练过程中的Tensorboard:

\* 初始化training(训练)

\* 运行下面的命令

\* 进入该网址查看详情:http://localhost:6006/

# logdir:log日志目录;port:端口

$ tensorboard --logdir='logs' --port=6006

**Train on Custom Dataset：(自定义训练集,ps:训练自己的模型)**

Custom model

运行下面的命令创建一个自定义模型，并且用我们自己类的矢量将<num-classes>代替了。

# 进入config目录

$ cd config/ # Navigate to config dir

# 下载自定义模型文件，下载后的文件名是yolov3-custom.cfg

$ bash create\_custom\_model.sh <num-classes> # Will create custom model 'yolov3-custom.cfg'

Classes

添加自定义的类名至data/custom/classes.name文件中。需要注意的是一个类名一行。

Image Folder

将数据集复制至data/custom/images目录中。

Annotation Folder

将我们标注好的文件放置data/custom/labels/目录中。这个数据加载文件希望与data/custom/iamges/train.jpg中对应，并且每个图片的文件名在data/custom/labels/train.txt文本文件中。注意在标注文件中每行中应该只定义一个bounding\_box，且每行的参数都只有label\_idx、x\_center、y\_center、width、height。这里的坐标应该被规定在[0,1]范围内并且label\_idx应该是0索引值并且每行都能与data/custom/classes.name中的类名保持一致。

Define Train and Validation Sets

在data/custom/train.txt和data/custom/valid.txt中，分别用作训练和验证数据的图像添加路径。

Train

训练自定义数据集，运行：

# model\_def:神经网络框架配置;data\_config:训练集目录

$ python3 train.py --model\_def config/yolov3-custom.cfg --data\_config config/custom.data

添加--pretrained\_weights weights/darknet53.conv.74可以使用在ImageNet中预训练的后端进行训练。

**Credit**

**YOLOv3: An Incremental Improvement**

Joseph Redmon, Ali Farhadi

Abstract:(摘要)

**附录一：**

|  |
| --- |
| numpy |
| torch>=1.0 |
| torchvision |
| matplotlib |
| tensorflow |
| tensorboard |
| terminaltables |
| pillow |
| tqdm |

特别需要注意的是：这个项目中不知运用了pytorch还使用了tensorflow。具体使用在了PyTorch-YOLOv3/utils/logger.py文件中。