

YOLOv5训练自定义模型

YOLOv5训练自定义模型

一、安装Pytorch 及 YOLO v5

1.1 安装pytorch GPU版

1.1.1 准备工作

1.1.2 安装pytorch

1.2 安装YOLO v5

二、YOLO v5训练自定义数据

2.1 准备数据集

2.1.1 创建 dataset.yaml

2.1.2 标注图片

2.1.3 组织目录结构

2.2 选择合适的预训练模型

2.3 训练

2.4 可视化

2.4.1 wandb

2.4.2 Tensorboard

2.3 测试评估模型

2.3.1 测试

2.3.2 评估

三、得到最优的训练结果

3.1 数据：

3.2 模型选择

3.3 训练

版本	作者	更新时间
V 1.0	@恩培-计算机视觉（抖音）	2022-03-31

说明：

- 1、训练过程请参考官网：<https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data>
- 2、本课使用的是YOLOv5 6.1版本，其他版本训练过程可能有不同，请以官网为准
- 3、硬件：Windows 10 、GPU GeForce 1060（6G）

一、安装Pytorch 及 YOLO v5

1.1 安装pytorch GPU版

1.1.1 准备工作

- 先去[pytorch官网](#)查看支持的CUDA版本：
 - 建议配合TensorFlow官网一起参考，以便两个库都可以使用
 - pytorch
 - 最新版：<https://pytorch.org/get-started/locally/>
 - 历史版本：<https://pytorch.org/get-started/previous-versions/>

PyTorch Build	Stable (1.11.0)		Preview (Nightly)	LTS (1.8.2)
Your OS	Linux		Mac	Windows
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.3	ROCm 4.2 (beta)	CPU
Run this Command:	conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch			

© 张松海-计算机视觉

- Tensorflow :
 - GPU支持CUDA列表：https://www.tensorflow.org/install/source_windows?hl=zh-cn

GPU

版本	Python 版本	编译器	构建工具	cuDNN	CUDA
tensorflow_gpu-2.6.0	3.6-3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.5.0	3.6-3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.4.0	3.6-3.8	MSVC 2019	Bazel 3.1.0	8.0	11.0
tensorflow_gpu-2.3.0	3.5-3.8	MSVC 2019	Bazel 3.1.0	7.6	10.1
tensorflow_gpu-2.2.0	3.5-3.8	MSVC 2019	Bazel 2.0.0	7.6	10.1
tensorflow_gpu-2.1.0	3.5-3.7	MSVC 2019	Bazel 0.27.1-0.29.1	7.6	10.1
tensorflow_gpu-2.0.0	3.5-3.7	MSVC 2017	Bazel 0.26.1	7.4	10
tensorflow_gpu-1.15.0	3.5-3.7	MSVC 2017	Bazel 0.26.1	7.4	10
tensorflow_gpu-1.14.0	3.5-3.7	MSVC 2017	Bazel 0.24.1-0.25.2	7.4	10
tensorflow_gpu-1.13.0	3.5-3.7	MSVC 2015 update 3	Bazel 0.19.0-0.21.0	7.4	10
tensorflow_gpu-1.12.0	3.5-3.6	MSVC 2015 update 3	Bazel 0.15.0	7.2	9.0
tensorflow_gpu-1.11.0	3.5-3.6	MSVC 2015 update 3	Bazel 0.15.0	7	9
tensorflow_gpu-1.10.0	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	7	9
tensorflow_gpu-1.9.0	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	7	9
tensorflow_gpu-1.8.0	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	7	9

© 张松海-计算机视觉

- 再查看所需CUDA版本对应的显卡驱动版本：

○ 参考信息：

- <https://docs.nvidia.com/deploy/cuda-compatibility/index.html#abstract>
- <https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html>
- <https://tech.amikeline.com/node-930/cuda-compatibility-of-nvidia-display-gpu-drivers/>

Search:

CUDA VERSION	COMPATIBLE DRIVER VERSION	REMARK	FIRST RELEASE
CUDA 11.6	>=450.80	Reference: CUDA toolkit release notes	January 2022
CUDA 11.5	>=450.80		October 2021
CUDA 11.4	>=450.80		June 2021
CUDA 11.3	>=450.80		April 2021
CUDA 11.2	>=450.80		December 2020
CUDA 11.1	>=450.80		September 2020
CUDA 11.0	>=450.36		March 2020
CUDA 10.2	>=440.33		November 2019
CUDA 10.1	>=418.39		February 2019
CUDA 10.0	>= 410.48		September 2018
CUDA 9.2 update 1	>= 396.37		June 2018
CUDA 9.2	>= 396.26		May 2018
CUDA 9.1	>= 390.46		December 2017
CUDA 9.0	>= 384.81		September 2017
CUDA 8.0 GA2	>= 375.26		February 2017
CUDA 8.0	>= 367.48		September 2016

© 绿蛙-计算数据

● 下载显卡对应版本驱动：

- 最新版： <https://www.nvidia.com/download/index.aspx>

NVIDIA Driver Downloads

Select from the dropdown list below to identify the appropriate driver for your NVIDIA product.

Product Type: GeForce
Product Series: GeForce 10 Series
Product: GeForce GTX 1060
Operating System: Windows 10 64-bit
Download Type: Game Ready Driver (GRD)
Language: English (US)

SEARCH

© 2018 NVIDIA Corporation

- 其他历史版本: <https://www.nvidia.com/Download/Find.aspx>

NVIDIA Driver Downloads

Official Advanced Driver Search | NVIDIA

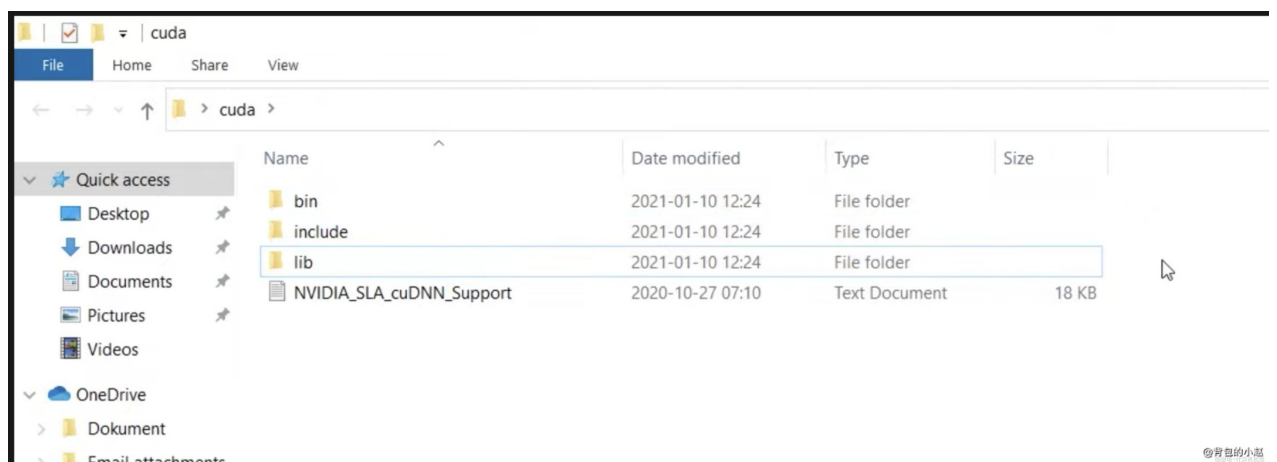
Product Type: GeForce
Product Series: GeForce 10 Series
Product: GeForce GTX 1060
Operating System: Windows 10 32-bit
Language: English (US)
Recommended/Beta: All

SEARCH

Name	Version	Release Date
GeForce Game Ready Driver WHQL	391.35	March 27, 2018
GeForce Game Ready Driver WHQL	391.24	March 20, 2018
GeForce Game Ready Driver WHQL	391.01	February 26, 2018
GeForce Game Ready Driver WHQL	390.77	January 29, 2018
GeForce Game Ready Driver WHQL	390.65	January 8, 2018
GeForce Game Ready Driver WHQL	388.71	December 20, 2017
GeForce Game Ready Driver WHQL	388.59	December 7, 2017
GeForce Game Ready Driver WHQL	388.43	November 30, 2017

© 2018 NVIDIA Corporation

- 下载对应版本CUDA:
 - 官网: <https://developer.nvidia.com/cuda-toolkit-archive>



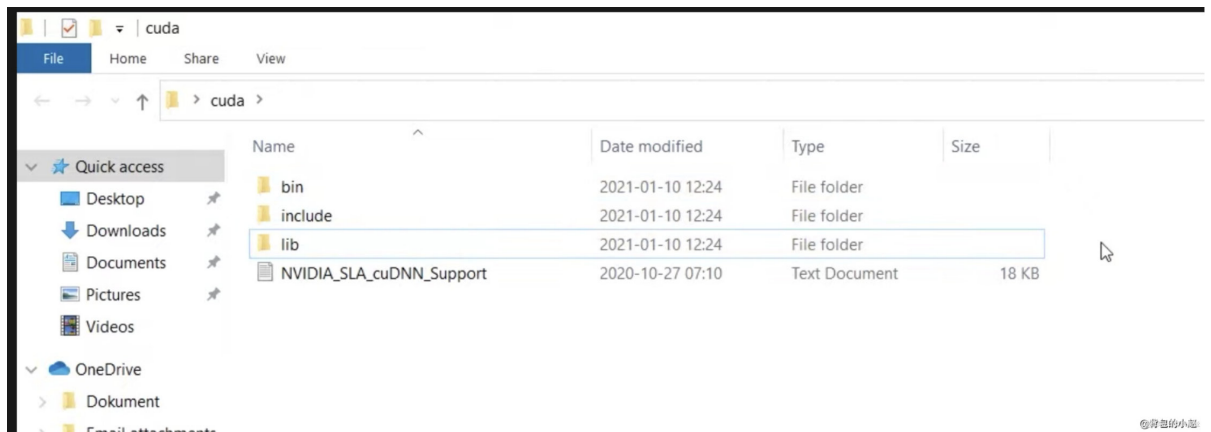
© 2018 NVIDIA Corporation

- 下载对应版本cuDNN
 - 参考TensorFlow GPU支持CUDA列表: https://www.tensorflow.org/install/source_windows?hl=zh-cn

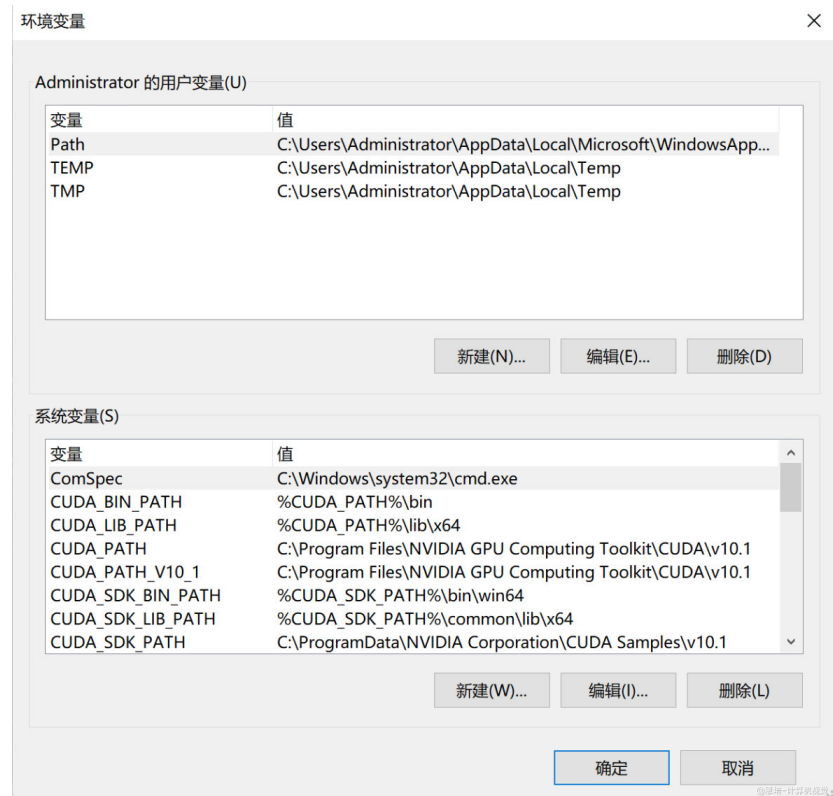
- cudnn官网: <https://developer.nvidia.com/zh-cn/cudnn>
- 下载VS studio: <https://visualstudio.microsoft.com/zh-hans/>



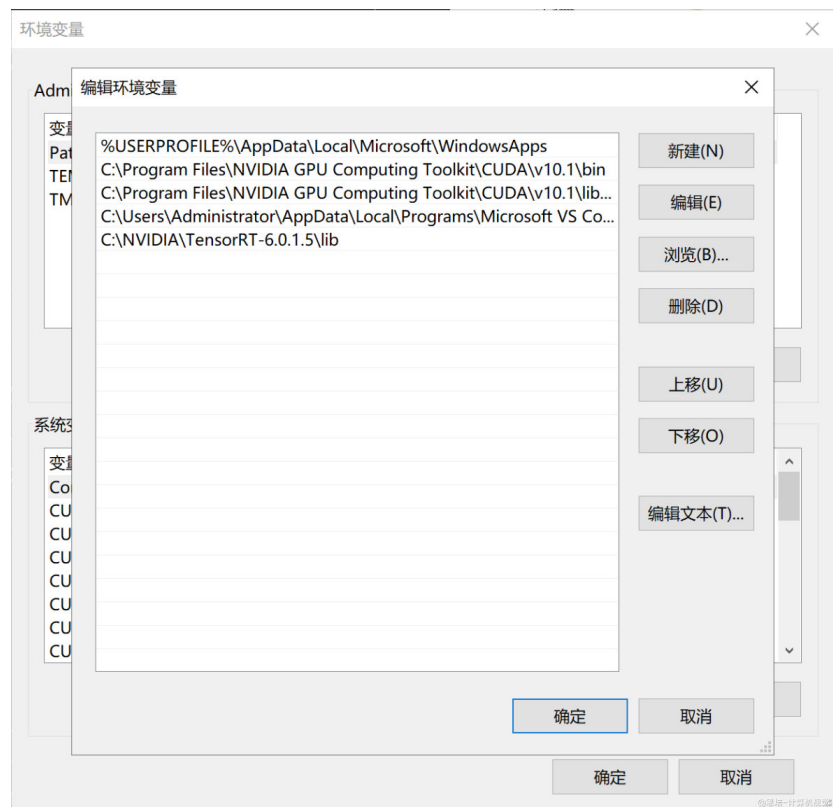
- 安装顺序:
 - VS studio: 安装社区版即可
 - 显卡驱动: 安装完重启电脑可以使用 `nvidia-smi` 查看显卡信息
 - CUDA: 按流程安装即可
 - cudnn:
 - 解压cudnn压缩文件:



- 进入cuda目录, 将cudnn所有文件复制并替换
 - 如我的cuda目录位置为: `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1`
- 更改环境变量:
 - 双击path



- 新建2个路径 (cuda bin、libnvvp)
 - 如我的路径为: `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin` 和 `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\libnvvp`



- 重启电脑

1.1.2 安装pytorch

创建conda虚拟环境，参考你选择的版本安装即可

- 最新版: <https://pytorch.org/get-started/locally/>
- 历史版本: <https://pytorch.org/get-started/previous-versions/>

如我的是 `pip install torch==1.8.1+cu101 torchvision==0.9.1+cu101 torchaudio==0.8.1 -f https://download.pytorch.org/whl/torch_stable.html`

1.2 安装YOLO v5













- 安装

```
# 克隆地址
git clone https://github.com/ultralytics/yolov5.git
# 如果下载速度慢，参考课程附件：
Windows软件/yolov5-master.zip

# 进入目录
cd yolov5
# 安装依赖
pip3 install -r requirements.txt
```

- 下载预训练权重文件

下载地址: <https://github.com/ultralytics/yolov5/releases>, 附件位置: `Windows软件/yolov5s.pt`, 将权重文件放到 `weights` 目录下:

▼Assets 12		
	yolov5l.pt	89.3 MB
	yolov5l6.pt	147 MB
	yolov5m.pt	40.8 MB
	yolov5m6.pt	69 MB
	yolov5n.pt	3.87 MB
	yolov5n6.pt	6.86 MB
	yolov5s.pt	14.1 MB
	yolov5s6.pt	24.8 MB
	yolov5x.pt	166 MB
	yolov5x6.pt	270 MB
	Source code (zip)	
	Source code (tar.gz)	

- 测试安装

```
python detect.py --source ./data/images/ --weights weights/yolov5s.pt --conf-thres 0.4
```

二、YOLO v5训练自定义数据

2.1 准备数据集

2.1.1 创建 dataset.yaml

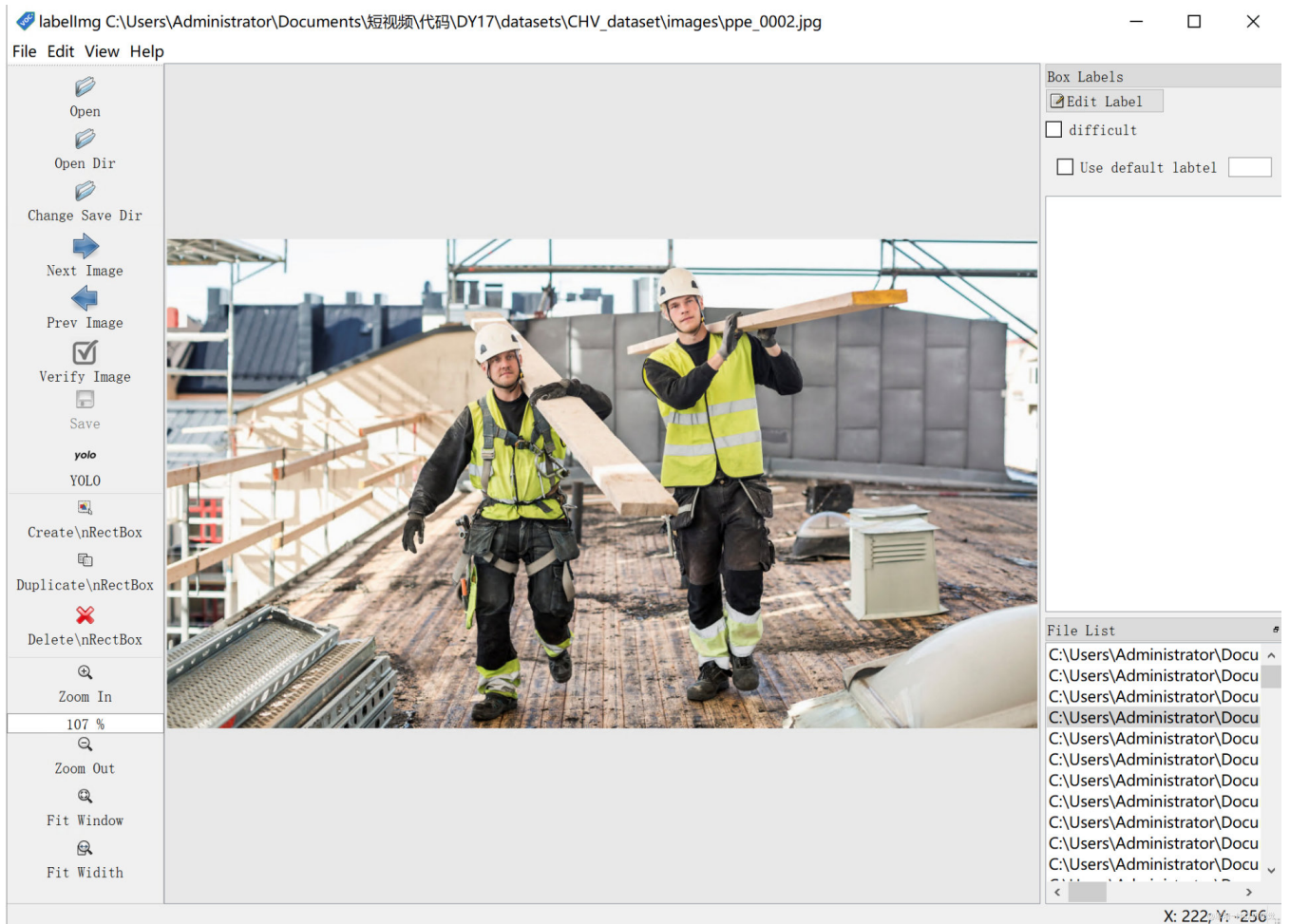
复制 `yolov5/data/coco128.yaml` 一份，比如为 `coco_chv.yaml`

```
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list:
[path/to/imgs1, path/to/imgs2, ..]
path: ../datasets/CHV_dataset # 数据所在目录
train: images/train # 训练集图片所在位置（相对于path）
val: images/val # 验证集图片所在位置（相对于path）
test: # 测试集图片所在位置（相对于path）（可选）

# 类别
nc: 6 # 类别数量
names: ['person', 'vest', 'blue helmet', 'red helmet', 'white helmet', 'yellow helmet'] # 类别标签名
```

2.1.2 标注图片

使用 [LabelImg](#) 等标注工具（需要支持YOLO格式，LabelImg附件路径：`Windows软件/labelImg.zip`）标注图片：



YOLO格式标签：

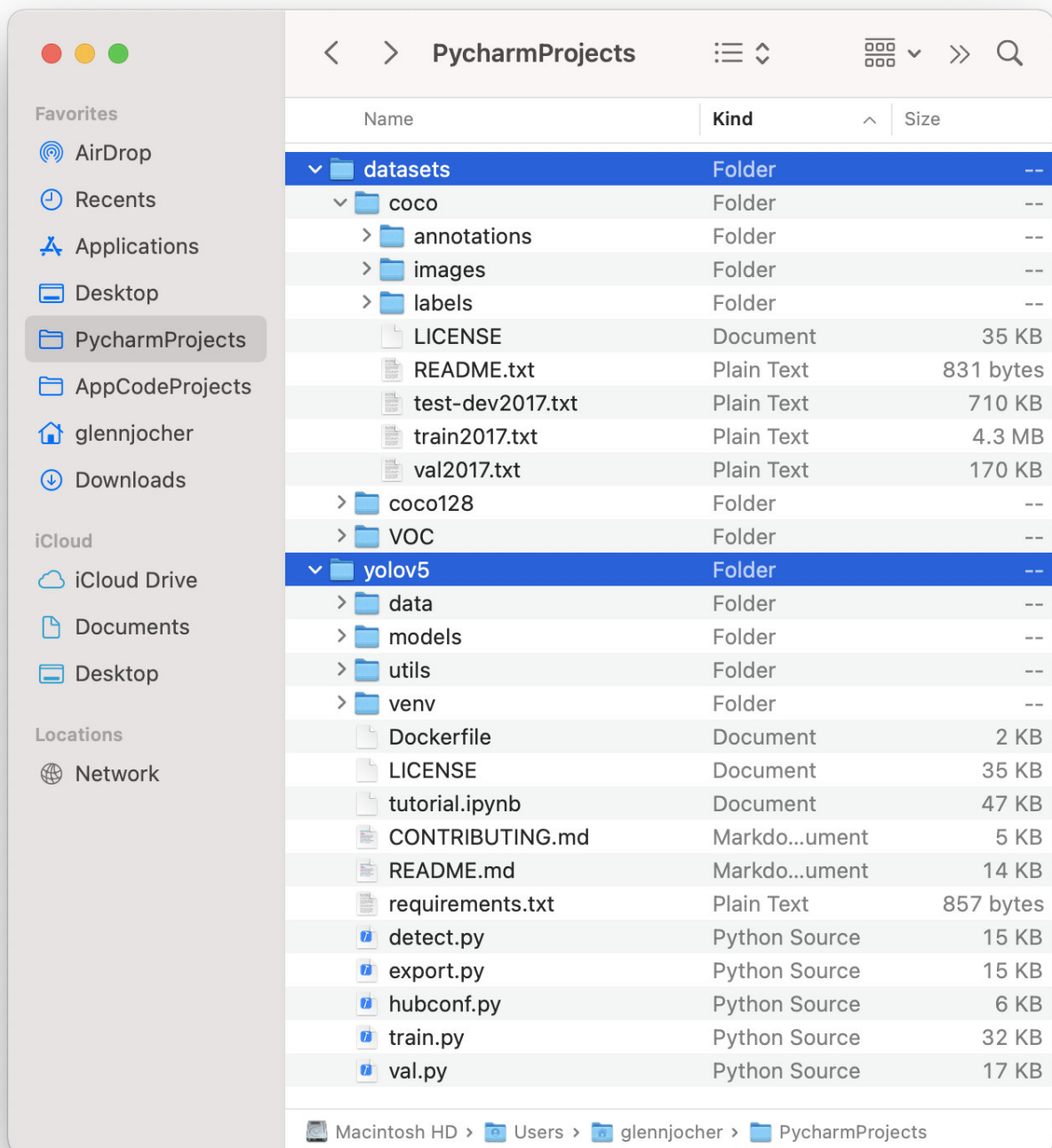




@何培-计算机视觉

- 一个图一个txt标注文件（如果图中无所要物体，则无需txt文件）；
- 每行一个物体；
- 每行数据格式：类别id、x_center y_center width height；
- **xywh**必须归一化（0-1），其中x_center、width除以图片宽度，y_center、height除以画面高度；
- 类别id必须从0开始计数。

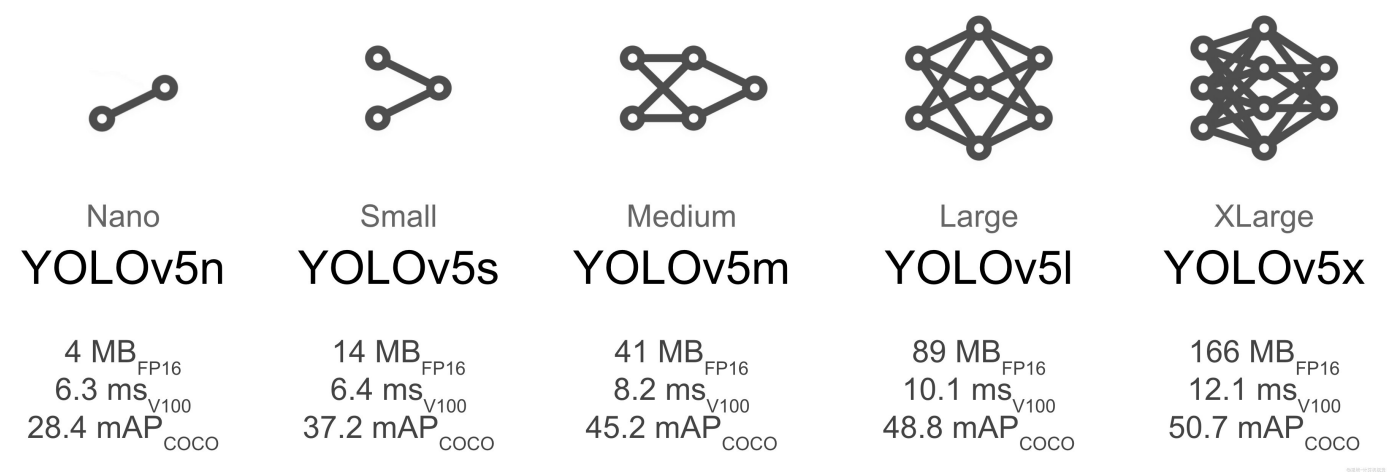
2.1.3 组织目录结构



@墨培-计算机视觉

- `datasets` 与 `yolov5` 同级目录；
- YOLO会自动将 `../datasets/CHV_dataset/images/train/ppe_1106.jpg` 中的 `/images/` 替换成 `/labels/` 以寻找它的标签，如 `../datasets/CHV_dataset/labels/train/ppe_1106.txt`，所以根据这个原则，我们一般可以：
 - `images` 文件夹下有 `train` 和 `val` 文件夹，分别放置训练集和验证集图片；
 - `labels` 文件夹有 `train` 和 `val` 文件夹，分别放置训练集和验证集标签(yolo格式)；

2.2 选择合适的预训练模型



根据你的设备，选择合适的预训练模型，具体模型比对如下：

Model	size (pixels)	mAP ^{val} 0.5:0.95	mAP ^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 55.8	72.7 72.7	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

复制 `models` 下对应模型的 `yaml` 文件，重命名，并修改其中：

```
nc: 80 # 类别数量
```

2.3 训练

下载对应的预训练模型权重文件，可以放到 `weights` 目录下，设置本机最好性能的各个参数，即可开始训练，课程中训练了以下参数：

```
# yolov5n
```

```
python .\train.py --data .\data\coco_chv.yaml --cfg .\models\yolov5n_chv.yaml --weights
.\weights\yolov5n.pt --batch-size 20 --epochs 120 --workers 4 --name base_n --project
yolo_test

# yolov5s
python .\train.py --data .\data\coco_chv.yaml --cfg .\models\yolov5s_chv.yaml --weights
.\weights\yolov5s.pt --batch-size 16 --epochs 120 --workers 4 --name base_s --project
yolo_test

# yolov5m
python .\train.py --data .\data\coco_chv.yaml --cfg .\models\yolov5m_chv.yaml --weights
.\weights\yolov5m.pt --batch-size 12 --epochs 120 --workers 4 --name base_m --project
yolo_test

# yolov5n6 1280
python .\train.py --data .\data\coco_chv.yaml --img-size 1280 --cfg
.\models\yolov5n6_chv.yaml --weights .\weights\yolov5n6.pt --batch-size 20 --epochs 120
--workers 4 --name base_n6 --project yolo_test
```

更多参数见 `train.py`;

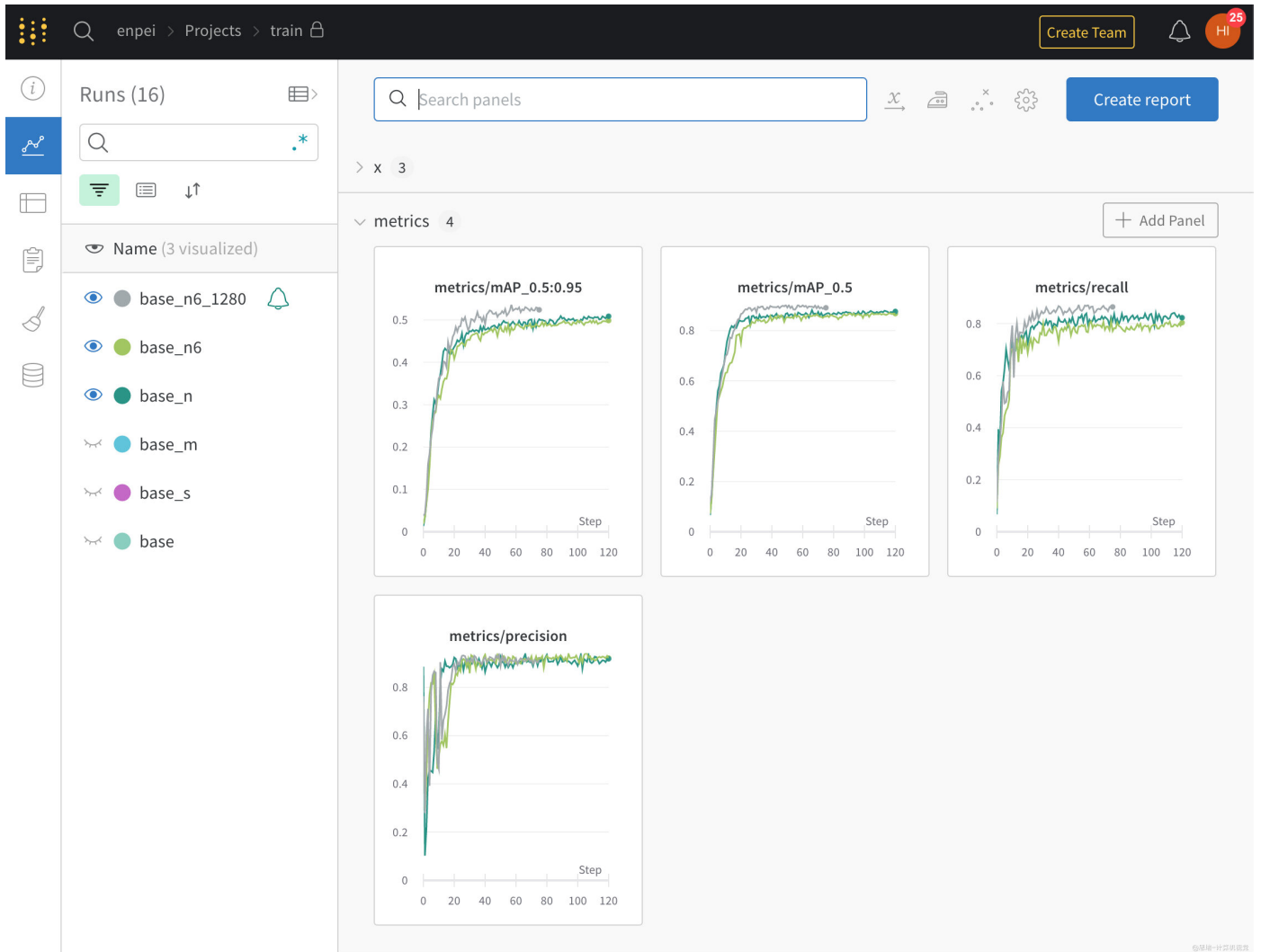
训练结果在 `runs/train/` 中可见，一般训练时间在几个小时以上。

2.4 可视化

2.4.1 wandb

YOLO官网推荐使用<https://wandb.ai/>。

- 使用 `pip install wandb` 安装扩展包;
- 去官网注册账号;
- 训练的时候填写 `key` 秘钥，地址: <https://wandb.ai/authorize>
- 打开网站即可查看训练进展。



2.4.2 Tensorboard

```
tensorboard --logdir=./runs
```

2.3 测试评估模型

2.3.1 测试

Usage - sources:

```
$ python path/to/detect.py --weights yolov5s.pt --source 0          # webcam
                                     img.jpg                         # image
                                     vid.mp4                         # video
                                     path/                           # directory
                                     path/*.jpg                      # glob
```

```
'https://youtu.be/Zgi9g1ksQHc' # YouTube
```

```
'rtsp://example.com/media.mp4' # RTSP, RTMP, HTTP stream
```

如

```
python detect.py --source ./test_img/img1.jpg --weights
runs/train/base_n/weights/best.pt --conf-thres 0.3
```

或

```
python detect.py --source 0 --weights runs/train/base_n/weights/best.pt --conf-thres
0.3
```

2.3.2 评估

n

```
# python val.py --data ./data/coco_chv.yaml --weights
runs/train/base_n/weights/best.pt --batch-size 12
```

4.3 GFLOPs

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	133	1084	0.88	0.823	0.868	0.479
person	133	450	0.899	0.808	0.877	0.484
vest	133	217	0.905	0.788	0.833	0.468
blue helmet	133	44	0.811	0.75	0.803	0.489
red helmet	133	50	0.865	0.9	0.898	0.425
white helmet	133	176	0.877	0.807	0.883	0.467
yellow helmet	133	147	0.922	0.885	0.917	0.543

Speed: 0.2ms pre-process, 4.7ms inference, 3.9ms NMS per image at shape (12, 3, 640, 640)

s

```
# python val.py --data ./data/coco_chv.yaml --weights
runs/train/base_s/weights/best.pt --batch-size 12
```

15.8 GFLOPs

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	133	1084	0.894	0.848	0.883	0.496
person	133	450	0.915	0.84	0.887	0.508
vest	133	217	0.928	0.834	0.877	0.501
blue helmet	133	44	0.831	0.75	0.791	0.428
red helmet	133	50	0.9	0.899	0.901	0.473


```

    white helmet      133      176      0.884      0.858      0.91      0.496
    yellow helmet     133      147      0.908      0.905      0.93      0.567
Speed: 0.2ms pre-process, 8.3ms inference, 3.9ms NMS per image at shape (12, 3, 640,
640)

```

```
# m
```

```
# python val.py --data ./data/coco_chv.yaml --weights
runs/train/base_m/weights/best.pt --batch-size 12
# 48.0 GFLOPs
```

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	133	1084	0.928	0.845	0.886	0.512
person	133	450	0.935	0.794	0.895	0.529
vest	133	217	0.922	0.813	0.868	0.508
blue helmet	133	44	0.916	0.818	0.812	0.464
red helmet	133	50	0.9	0.9	0.892	0.488
white helmet	133	176	0.932	0.841	0.899	0.511
yellow helmet	133	147	0.964	0.905	0.948	0.574

```

Speed: 0.4ms pre-process, 18.8ms inference, 4.6ms NMS per image at shape (12, 3, 640,
640)

```

```
# n6 1280 :
```

```
# python val.py --data ./data/coco_chv.yaml --weights
runs/train/base_n6_1280/weights/best.pt --batch-size 12 --img-size 1280
# 4.3 GFLOPs
```

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	133	1084	0.906	0.858	0.901	0.507
person	133	450	0.903	0.831	0.887	0.503
vest	133	217	0.922	0.816	0.86	0.486
blue helmet	133	44	0.843	0.795	0.828	0.465
red helmet	133	50	0.899	0.92	0.954	0.507
white helmet	133	176	0.921	0.865	0.925	0.515
yellow helmet	133	147	0.947	0.918	0.954	0.566

```

Speed: 1.5ms pre-process, 14.1ms inference, 2.2ms NMS per image at shape (12, 3, 1280,
1280)

```





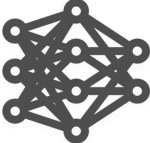
三、得到最优的训练结果

参考: <https://github.com/ultralytics/yolov5/wiki/Tips-for-Best-Training-Results>

3.1 数据：

- 每类图片：建议 ≥ 1500 张；
- 每类实例（标注的物体）：建议 ≥ 10000 个；
- 图片采样：真实图片建议在一天中不同时间、不同季节、不同天气、不同光照、不同角度、不同来源（爬虫抓取、手动采集、不同相机源）等场景下采集；
- 标注：
 - 所有图片上所有类别的对应物体都需要标注上，不可以只标注部分；
 - 标注尽量闭合物体，边界框与物体无空隙，所有类别对应物体不能缺少标签；
- 背景图：背景图用于减少假阳性预测（False Positive），建议提供0~10%样本总量的背景图，背景图无需标注；

3.2 模型选择

				
Nano	Small	Medium	Large	XLarge
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
4 MB _{FP16} 6.3 ms _{V100} 28.4 mAP _{COCO}	14 MB _{FP16} 6.4 ms _{V100} 37.2 mAP _{COCO}	41 MB _{FP16} 8.2 ms _{V100} 45.2 mAP _{COCO}	89 MB _{FP16} 10.1 ms _{V100} 48.8 mAP _{COCO}	166 MB _{FP16} 12.1 ms _{V100} 50.7 mAP _{COCO}

模型越大一般预测结果越好，但相应的计算量越大，训练和运行起来都会慢一点，建议：

- 在移动端（手机、嵌入式）选择：YOLOv5n/s/m
- 云端（服务器）选择：YOLOv5l/x

3.3 训练

- 对于小样本、中样本，建议试用预训练模型开始训练：

```
python train.py --data custom.yaml --weights yolov5s.pt
                                                    yolov5m.pt
                                                    yolov5l.pt
                                                    yolov5x.pt
                                                    custom_pretrained.pt
```

- 对于大样本，建议从0开始训练（无需预训练模型）：

```
# --weights ''
```

```
python train.py --data custom.yaml --weights '' --cfg yolov5s.yaml  
yolov5m.yaml  
yolov5l.yaml  
yolov5x.yaml
```

- Epochs: 初始设定为300, 如果很早就过拟合, 减少epoch, 如果到300还没过拟合, 设置更大的数值, 如600, 1200等;
- 图像尺寸: 训练时默认为 `--img 640`, 如果希望检测出画面中的小目标, 可以设为 `--img 1280` (检测时也需要设为 `--img 1280` 才能起到一样的效果)
- Batch size: 选择你硬件能承受的最大 `--batch-size`;
- 超参数 (**Hyperparameters**): 初次训练暂时不要改, 具体参见<https://github.com/ultralytics/yolov5/issues/607>
- 更多: 官网建议 查看<http://karpathy.github.io/2019/04/25/recipe/>