海思3516移植YOLOV5步骤

# [Hi3516上yolov5的pytorch转onnx转caffe转wk详解](https://www.cnblogs.com/d442130165/p/16711189.html" \o "发布于 2022-09-20 15:33)

1. **环境配置**

VMware Workstation Pro 16.2.4

Ubuntu 18.04.6

Pytorch 1.10.0

Caffe-cpu 1.0

Python 3.6

Opencv 3.2.0

2. **下载并安装虚拟机**

下载地址：<https://customerconnect.vmware.com/cn/downloads/details?downloadGroup=WKST-1624-WIN&productId=1038&rPId=91434>

安装秘钥：

ZF3R0-FHED2-M80TY-8QYGC-NPKYF

3. **下载并安装Ubuntu18.04**

下载地址：<https://releases.ubuntu.com/18.04/ubuntu-18.04.6-desktop-amd64.iso>

4. **安装Pytorch1.10.0，注意ubuntu有默认Python为Python2，下面均使用Python3**

5.1修改pip3源

sudo apt update

sudo apt upgrade

sudo apt install vim

sudo apt install python3-pip

sudo pip3 install --upgrade pip

修改源

mkdir ~/.pip

vim ~/.pip/pip.conf

[global]

index-url = https://pypi.tuna.tsinghua.edu.cn/simple

或者

pip config set global.index-url https://pypi.mirrors.ustc.edu.cn/simple/

vim .config/pip/pip.conf 或 pip config list

可选安装pytorch，或者在yolov5的reqirements.txt一起安装

pip3 install torch==1.10.0+cpu torchvision==0.11.0+cpu torchaudio==0.10.0 -f https://download.pytorch.org/whl/torch\_stable.html

5. **下载YOLOv5-4.0，修改网络结构，进行训练，也可下载yolov5-6.0**

6.1下载Yolov5-4.0

下载地址：<https://codeload.github.com/ultralytics/yolov5/zip/refs/tags/v4.0>

6.2安装依赖文件

cd yolov5-4.0

pip3 install -r requirements.txt

6.3 添加数据集和配置文件

cd yolov5-4.0

vim data/roadsign\_voc.yaml

train: ../data/train.txt # train images (relative to 'path') 128 images

val: ../data/valid.txt # val images (relative to 'path') 128 images

test: # test images (optional)

# Classes

nc: 4 # number of classes

names: ['speedlimit', 'crosswalk', 'trafficlight', 'stop'] # class names

[记得这里要自己手动创建data和vaild.txt，然后修改其中的路径为绝对路径，两个txt里面也是绝对路径去寻找照片，标红部分指需要修改路径]

6.4 修改网络配置

cd yolov5-4.0

vim models/yolov5s.yaml

[IMG_256](javascript:void(0);)

# parameters

nc: 80 # 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

# YOLOv5 backbone

backbone:

# [from, number, module, args]

[

#[-1, 1, Focus, [64, 3]], # 0-P1/2

[-1, 1, Conv, [64, 3,2]], # 0-P1/2

[-1, 1, Conv, [128, 3, 2]], # 1-P2/4

[-1, 3, C3, [128]],

[-1, 1, Conv, [256, 3, 2]], # 3-P3/8

[-1, 9, C3, [256]],

[-1, 1, Conv, [512, 3, 2]], # 5-P4/16

[-1, 9, C3, [512]],

[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32

[-1, 1, SPP, [1024, [5, 9, 13]]],

[-1, 3, C3, [1024, False]], # 9

]

# YOLOv5 headhead:

[[-1, 1, Conv, [512, 1, 1]],

#[-1, 1, nn.Upsample, [None, 2, 'nearest']],

[-1, 1, nn.ConvTranspose2d, [256, 256, 2,2]],

[[-1, 6], 1, Concat, [1]], # cat backbone P4

[-1, 3, C3, [512, False]], # 13

[-1, 1, Conv, [256, 1, 1]],

#[-1, 1, nn.Upsample, [None, 2, 'nearest']],

[-1, 1, nn.ConvTranspose2d, [128, 128, 2,2]],

[[-1, 4], 1, Concat, [1]], # cat backbone P3

[-1, 3, C3, [256, False]], # 17 (P3/8-small)

[-1, 1, Conv, [256, 3, 2]],

[[-1, 14], 1, Concat, [1]], # cat head P4

[-1, 3, C3, [512, False]], # 20 (P4/16-medium)

[-1, 1, Conv, [512, 3, 2]],

[[-1, 10], 1, Concat, [1]], # cat head P5

[-1, 3, C3, [1024, False]], # 23 (P5/32-large)

[[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)

]

[IMG_257](javascript:void(0);)

6.5 修改网络结构

cd yolov5-4.0

vim models/common.py

[IMG_258](javascript:void(0);)

class Conv(nn.Module):

# Standard convolution

def \_\_init\_\_(self, c1, c2, k=1, s=1, p=None, g=1, act=True): # ch\_in, ch\_out, kernel, stride, padding, groups

super(Conv, self).\_\_init\_\_()

self.conv = nn.Conv2d(c1, c2, k, s, autopad(k, p), groups=g, bias=False)

self.bn = nn.BatchNorm2d(c2)

# self.act = nn.SiLU() if act is True else (act if isinstance(act, nn.Module) else nn.Identity())

self.act = nn.ReLU() if act is True else (act if isinstance(act, nn.Module) else nn.Identity())

def forward(self, x):

return self.act(self.bn(self.conv(x)))

def fuseforward(self, x):

return self.act(self.conv(x))

class BottleneckCSP(nn.Module):

# CSP Bottleneck https://github.com/WongKinYiu/CrossStagePartialNetworks

def \_\_init\_\_(self, c1, c2, n=1, shortcut=True, g=1, e=0.5): # ch\_in, ch\_out, number, shortcut, groups, expansion

super(BottleneckCSP, self).\_\_init\_\_()

c\_ = int(c2 \* e) # hidden channels

self.cv1 = Conv(c1, c\_, 1, 1)

self.cv2 = nn.Conv2d(c1, c\_, 1, 1, bias=False)

self.cv3 = nn.Conv2d(c\_, c\_, 1, 1, bias=False)

self.cv4 = Conv(2 \* c\_, c2, 1, 1)

self.bn = nn.BatchNorm2d(2 \* c\_) # applied to cat(cv2, cv3)

# self.act = nn.LeakyReLU(0.1, inplace=True)

self.act = nn.ReLU()

self.m = nn.Sequential(\*[Bottleneck(c\_, c\_, shortcut, g, e=1.0) for \_ in range(n)])

def forward(self, x):

y1 = self.cv3(self.m(self.cv1(x)))

y2 = self.cv2(x)

return self.cv4(self.act(self.bn(torch.cat((y1, y2), dim=1))))

class SPP(nn.Module):

# Spatial pyramid pooling layer used in YOLOv3-SPP

def \_\_init\_\_(self, c1, c2, k=(5, 9, 13)):

super(SPP, self).\_\_init\_\_()

c\_ = c1 // 2 # hidden channels

self.cv1 = Conv(c1, c\_, 1, 1)

self.cv2 = Conv(c\_ \* (len(k) + 1), c2, 1, 1)

# self.m = nn.ModuleList([nn.MaxPool2d(kernel\_size=x, stride=1, padding=x // 2) for x in k])

self.m = nn.ModuleList([nn.MaxPool2d(kernel\_size=x, stride=1, padding=x // 2, ceil\_mode=True) for x in k])

def forward(self, x):

x = self.cv1(x)

return self.cv2(torch.cat([x] + [m(x) for m in self.m], 1))

[IMG_259](javascript:void(0);)

6.6 训练模型

cd yolov5-4.0/weights

wget [https://github.com/ultralytics/yolov5/releases/download/v4.0/yolov5s.pt](https://github.com/ultralytics/yolov5/releases/download/v3.1/yolov5s.pt)

python3 train.py --img 640 --batch 16 --epochs 1 --data data/roadsign\_voc.yaml --cfg models/yolov5s.yaml --weights weights/yolov5s.pt --noautoanchor

6. **导出onnx模型**

pip3 install onnx==1.8.1

pip3 install onnx-simplifier

cd yolov5-4.0/

vim models/export.py

model.model[-1].export = True

torch.onnx.export(model, img, f, verbose=False, opset\_version=10, input\_names=['images'], output\_names=['classes', 'boxes'] if y is None else ['output'])

python3 models/export.py --weights runs/train/exp/weights/last.pt

python3 -m onnxsim runs/train/exp/weights/last.onnx runs/train/exp/weights/yolov5s\_sim.onnx

7. **下载并安装yolov5\_caffe，编译生成caffe和pycaffe**

7.1下载地址：

<https://codeload.github.com/Wulingtian/yolov5_caffe/zip/refs/heads/master>

7.2安装依赖包

sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler

sudo apt-get install --no-install-recommends libboost-all-dev

sudo apt-get install python3-dev

sudo apt-get install libatlas-base-dev

sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev

sudo apt-get install python3-opencv

7.3修改安装配置文件，使用CPU版本进行编译安装caffe

cd yolov5\_caffe

**vim Makefile.config**

[IMG_260](javascript:void(0);)

# USE\_CUDNN := 1

CPU\_ONLY := 1

OPENCV\_VERSION := 3

BLAS := atlas

# CUDA\_DIR := /usr

# CUDA\_ARCH := -gencode arch=compute\_61,code=compute\_61

# ANACONDA\_HOME := $(HOME)/anaconda

# PYTHON\_INCLUDE := $(ANACONDA\_HOME)/include \

# $(ANACONDA\_HOME)/include/python3.6m \

# $(ANACONDA\_HOME)/lib/python3.6/site-packages/numpy/core/include

PYTHON\_LIBRARIES := boost\_python3 python3.6m

PYTHON\_INCLUDE := /usr/include/python3.6m \

/usr/lib/python3.6/dist-packages/numpy/core/include

#PYTHON\_LIBRARIES := boost\_python-py35

# PYTHON\_INCLUDE := /usr/include/python3.5m \

# /usr/lib/python3.5/dist-packages/numpy/core/include

PYTHON\_LIB := /usr/lib

# PYTHON\_LIB := $(ANACONDA\_HOME)/lib

WITH\_PYTHON\_LAYER := 1

INCLUDE\_DIRS := $(PYTHON\_INCLUDE) /usr/local/include /usr/include/hdf5/serial

LIBRARY\_DIRS := $(PYTHON\_LIB) /usr/local/lib /usr/lib /usr/lib/x86\_64-linux-gnu /usr/lib/x86\_64-linux-gnu/hdf5/serial

BUILD\_DIR := build

DISTRIBUTE\_DIR := distribute

# TEST\_GPUID := 0

Q ?= @

[IMG_261](javascript:void(0);)

**vim Makefile**

LIBRARIES += glog gflags protobuf boost\_system boost\_filesystem m hdf5\_serial\_hl hdf5\_serial

验证是否安装成功

make clean

make all -j8

make test -j8

make runtest -j8

7.4安装pycaffe

cd yolov5\_caffe/python

for req in (catrequirements.txt);dopipinstall(catrequirements.txt);dopipinstallreq; done

或者

sudo apt-get install python-numpy python-scipy python-matplotlib python-sklearn python-skimage python-h5py python-protobuf python-leveldb python-networkx python-nose python-pandas python-gflags Cython ipython

sudo apt-get install protobuf-c-compiler protobuf-compiler

cd yolov5\_caffe/

make pycaffe -j8

vim ~/.bashrc

export PYTHONPATH=.../yolov5\_caffe/python:$PYTHONPATH

source ~/.bashrc

验证是否安装成功

python3

import caffe

8. **下载yolov5\_onnx2caff，把onnx模型转换为caffe模型**

下载地址：

<https://codeload.github.com/Hiwyl/yolov5_onnx2caffe/zip/refs/heads/master>

cd yolov5\_onnx2caffe/

vim convertCaffe.py

onnx\_path = "./weights/yolov5s\_sim.onnx"

prototxt\_path = "./weights/yolov5s\_sim.prototxt"

caffemodel\_path = "./weights/yolov5s\_sim.caffemodel"

convertToCaffe(graph, prototxt\_path, caffemodel\_path)

9. **验证转换的caffe模型**

cd yolov5\_caffe/tools

设置输入参数和模型路径

vim caffe\_yolov5s.cpp

[IMG_262](javascript:void(0);)

#define INPUT\_W 640#define INPUT\_H 640#define IsPadding 1#define NUM\_CLASS 4#define NMS\_THRESH 0.6#define CONF\_THRESH 0.3

std::string prototxt\_path = "./weights/yolov5s\_sim.prototxt";

std::string caffemodel\_path = "./weithgts/yolov5s\_sim.caffemodel";

std::string pic\_path = "./weights/road580.png";

[IMG_263](javascript:void(0);)

编译caffe\_yolov5s

cd yolov5\_caffe/

sudo apt install cmake

make

运行caffe\_yolov5s

cp ../yolov5\_onnx2caffe/weights/yolov5s\_sim.caffemodel weights/

cp ../yolov5\_onnx2caffe/weights/yolov5s\_sim.prototxt weights/

./build/tools/caffe\_yolov5s

查看检测结果

xdg-open result.jpg

10. **安装nnie mapper，把caffe模型转换为wk模型**

10.1 在hi3516dv300中提供的资料中找到nnie\_mapper\_12，安装参考文档HiSVP 开发指南的 Linux 版 NNIE mapper 安装

nnie\_mapper\_12位置在SVP\_PC\HiSVP\_PC\_V1.2.2.2\tools\nnie\linux\mapper中

使用readelf查看nnie\_mapper\_12的依赖库，需要protobuf3.6.1和opencv3.4

readelf -d nnie\_mapper\_12

10.2 安装opencv3.4.2

下载地址：<https://github.com/opencv/opencv/archive/3.4.2.zip>

安装依赖

sudo apt-get install build-essential

sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libtiff-dev libjasper-dev libdc1394-22-dev

安装opencv

mkdir yolov5\_caffe2nnie

cd yolov5\_caffe2nnie

mkdir 3rd

cd 3rd

tar -xvzf opencv-3.4.2.tar.gz

cd opencv-3.4.2

mkdir build && cd build

cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=.../yolov5\_caffe2nnie ../

make -j8

make install

添加环境路径

vim ~/.bashrc

export PATH=.../yolov5\_caffe2nnie/bin:$PATH

export LD\_LIBRARY\_PATH=.../yolov5\_caffe2nnie/lib:$LD\_LIBRARY\_PATH

export PKG\_CONFIG\_PATH=.../yolov5\_caffe2nnie/lib/pkgconfig

source ~/.bashrc

验证opencv是否安装成功

运行opencv\_version显示3.4.2

编译opencv程序

sudo apt-get install libcanberra-gtk-module

cd opencv-3.4.2/samples/cpp/

 gcc `pkg-config --cflags opencv` -o facedetect facedetect.cpp `pkg-config --libs opencv` -lstdc++

./facedetect ../data/lena.jpg

10.3 安装protobuf3.6.1

protobuf3.6.1编译需要用gcc4.8，Ubuntu18.04自带gcc7.5，需要安装多版本gcc

sudo apt update

sudo apt install build-essential

sudo apt install software-properties-common

sudo add-apt-repository ppa:ubuntu-toolchain-r/test

sudo apt install gcc-4.8 g++-4.8 gcc-7 g++-7 gcc-8 g++-8 gcc-9 g++-9

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 48 --slave /usr/bin/g++ g++ /usr/bin/g++-4.8

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 70 --slave /usr/bin/g++ g++ /usr/bin/g++-7

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-8 80 --slave /usr/bin/g++ g++ /usr/bin/g++-8

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 --slave /usr/bin/g++ g++ /usr/bin/g++-9

切换到gcc4.8

sudo update-alternatives --config gcc

gcc -v

下载protobuf3.6.1：

<https://github.com/google/protobuf/releases/download/v3.6.1/protobuf-all-3.6.1.tar.gz>

tar -xvf protobuf.3.6.1

cd protobuf3.6.1

./configure -prefix=.../yolov5\_caffe2nnie

make

make check

make install

验证protobuf.3.6.1

protoc --version

显示libprotoc 3.6.1

切换到gcc7.5

sudo update-alternatives --config gcc

gcc -v

10.4 安装nnie\_mapper\_12

复制nnie\_mapper\_12到.../yolov5\_caffe2nnie/bin

验证nnie\_mapper\_12

复制SVP\_PC\HiSVP\_PC\_V1.2.2.2\software\data到.../yolov5\_caffe2nnie/

cd .../yolov5\_caffe2nnie/data

运行nnie\_mapper\_12 classification/alexnet/alexnet\_no\_group\_inst.cfg

显示下列信息，说明安装成功

[IMG_264](javascript:void(0);)

Mapper Version 1.2.2.1\_B030 (NNIE\_1.2) 19090610466402

begin net parsing....

end net parsing

begin prev optimizing....

end prev optimizing....

begin net quantalizing(CPU)....

end quantalizing

begin optimizing....

end optimizing

begin NNIE[0] mem allocation....

end NNIE[0] memory allocating

begin NNIE[0] instruction generating....

end NNIE[0] instruction generating

begin parameter compressing....

end parameter compressing

begin compress index generating....

end compress index generating

begin binary code generating....

end binary code generating

begin quant files writing....

end quant files writing

[IMG_265](javascript:void(0);)

10.5 修改yolov5的caffe网络结构，并转换为wk模型

复制onnx转换好的caffe模型到.../yolov5\_caffe2nnie/

cp ../yolov5\_onnx2caffe/weights/yolov5s\_sim.caffemodel ./weights/

cp ../yolov5\_onnx2caffe/weights/yolov5s\_sim.prototxt ./weights/

修改yolov5的caffe网络结构的三个输出头部

vim yolov5s\_sim.prototxt

删除三个Permute

修改三个Reshape为下面格式

[IMG_266](javascript:void(0);)

layer {

name: "Reshape\_151"

type: "Reshape"

bottom: "265"

top: "277"

reshape\_param {

shape {

dim: 0

dim: 3

dim: 9

dim: 6400

}

}

}

layer {

name: "Reshape\_165"

type: "Reshape"

bottom: "279"

top: "291"

reshape\_param {

shape {

dim: 0

dim: 3

dim: 9

dim: 1600

}

}

}

layer {

name: "Reshape\_179"

type: "Reshape"

bottom: "293"

top: "305"

reshape\_param {

shape {

dim: 0

dim: 3

dim: 9

dim: 400

}

}

}

[IMG_267](javascript:void(0);)

编写转换配置文件，参考yolov3的配置文件，具体参数参考HiSVP 开发指南中的nnie\_mapperde的配置文件说明

vim weights/yolov5s\_sim.cfg

[IMG_268](javascript:void(0);)

[prototxt\_file] ./weights/yolov5s\_sim.prototxt

[caffemodel\_file] ./weights/yolov5s\_sim.caffemodel

[batch\_num] 1

[net\_type] 0

[sparse\_rate] 0

[compile\_mode] 1

[is\_simulation] 0

[log\_level] 2

[instruction\_name] ./weights/yolov5s\_sim

[RGB\_order] BGR

[data\_scale] 0.0039062

[internal\_stride] 16

[image\_list] ../../data/valid.txt

[image\_type] 1

[mean\_file] null

[norm\_type] 3

[IMG_269](javascript:void(0);)

11. **下载YOLOv5-6.0，修改网络结构，进行训练**

11.1下载Yolov5-6.0

下载地址：<https://codeload.github.com/ultralytics/yolov5/zip/refs/tags/v4.0>

11.2安装依赖文件

cd yolov5-6.0

pip3 install -r requirements.txt

11.3 添加数据集和配置文件

cd yolov5-6.0

vim data/roadsign\_voc.yaml

train: ../data/train.txt  # train images (relative to 'path') 128 images

val: ../data/valid.txt  # val images (relative to 'path') 128 images

test:  # test images (optional)

# Classes

nc: 4  # number of classes

names: ['speedlimit', 'crosswalk', 'trafficlight', 'stop']   # class names

11.4 修改网络配置

cd yolov5-6.0

vim models/yolov5s.yaml

[IMG_270](javascript:void(0);)

# YOLOv5 v6.0 backbone

backbone:

# [from, number, module, args]

[[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2

[-1, 1, Conv, [128, 3, 2]], # 1-P2/4

[-1, 3, C3, [128]],

[-1, 1, Conv, [256, 3, 2]], # 3-P3/8

[-1, 6, C3, [256]],

[-1, 1, Conv, [512, 3, 2]], # 5-P4/16

[-1, 9, C3, [512]],

[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32

[-1, 3, C3, [1024]],

[-1, 1, SPPF, [1024, 5]], # 9

]

# YOLOv5 v6.0 headhead:

[[-1, 1, Conv, [512, 1, 1]],

#[-1, 1, nn.Upsample, [None, 2, 'nearest']],

[-1, 1, nn.ConvTranspose2d, [256, 256, 2, 2]],

[[-1, 6], 1, Concat, [1]], # cat backbone P4

[-1, 3, C3, [512, False]], # 13

[-1, 1, Conv, [256, 1, 1]],

#[-1, 1, nn.Upsample, [None, 2, 'nearest']],

[-1, 1, nn.ConvTranspose2d, [128, 128, 2, 2]],

[[-1, 4], 1, Concat, [1]], # cat backbone P3

[-1, 3, C3, [256, False]], # 17 (P3/8-small)

[-1, 1, Conv, [256, 3, 2]],

[[-1, 14], 1, Concat, [1]], # cat head P4

[-1, 3, C3, [512, False]], # 20 (P4/16-medium)

[-1, 1, Conv, [512, 3, 2]],

[[-1, 10], 1, Concat, [1]], # cat head P5

[-1, 3, C3, [1024, False]], # 23 (P5/32-large)

[[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)

]

[IMG_271](javascript:void(0);)

6.5 修改网络结构

cd yolov5-6.0

vim models/common.py

[IMG_272](javascript:void(0);)

class Conv(nn.Module):

# Standard convolution

def \_\_init\_\_(self, c1, c2, k=1, s=1, p=None, g=1, act=True): # ch\_in, ch\_out, kernel, stride, padding, groups

super().\_\_init\_\_()

self.conv = nn.Conv2d(c1, c2, k, s, autopad(k, p), groups=g, bias=False)

self.bn = nn.BatchNorm2d(c2)

#self.act = nn.SiLU() if act is True else (act if isinstance(act, nn.Module) else nn.Identity())

self.act = nn.ReLU() if act is True else (act if isinstance(act, nn.Module) else nn.Identity())

def forward(self, x):

return self.act(self.bn(self.conv(x)))

def forward\_fuse(self, x):

return self.act(self.conv(x))

lass BottleneckCSP(nn.Module):

# CSP Bottleneck https://github.com/WongKinYiu/CrossStagePartialNetworks

def \_\_init\_\_(self, c1, c2, n=1, shortcut=True, g=1, e=0.5): # ch\_in, ch\_out, number, shortcut, groups, expansion

super().\_\_init\_\_()

c\_ = int(c2 \* e) # hidden channels

self.cv1 = Conv(c1, c\_, 1, 1)

self.cv2 = nn.Conv2d(c1, c\_, 1, 1, bias=False)

self.cv3 = nn.Conv2d(c\_, c\_, 1, 1, bias=False)

self.cv4 = Conv(2 \* c\_, c2, 1, 1)

self.bn = nn.BatchNorm2d(2 \* c\_) # applied to cat(cv2, cv3)

#self.act = nn.LeakyReLU(0.1, inplace=True)

self.act = nn.ReLU()

self.m = nn.Sequential(\*[Bottleneck(c\_, c\_, shortcut, g, e=1.0) for \_ in range(n)])

def forward(self, x):

y1 = self.cv3(self.m(self.cv1(x)))

y2 = self.cv2(x)

return self.cv4(self.act(self.bn(torch.cat((y1, y2), dim=1))))

class SPPF(nn.Module):

# Spatial Pyramid Pooling - Fast (SPPF) layer for YOLOv5 by Glenn Jocher

def \_\_init\_\_(self, c1, c2, k=5): # equivalent to SPP(k=(5, 9, 13))

super().\_\_init\_\_()

c\_ = c1 // 2 # hidden channels

self.cv1 = Conv(c1, c\_, 1, 1)

self.cv2 = Conv(c\_ \* 4, c2, 1, 1)

#self.m = nn.MaxPool2d(kernel\_size=k, stride=1, padding=k // 2)

self.m = nn.MaxPool2d(kernel\_size=k, stride=1, padding=k // 2, ceil\_mode=True)

def forward(self, x):

x = self.cv1(x)

with warnings.catch\_warnings():

warnings.simplefilter('ignore') # suppress torch 1.9.0 max\_pool2d() warning

y1 = self.m(x)

y2 = self.m(y1)

return self.cv2(torch.cat([x, y1, y2, self.m(y2)], 1))

[IMG_273](javascript:void(0);)

11.6 训练模型

cd yolov5-6.0/weights

wget [https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt](https://github.com/ultralytics/yolov5/releases/download/v3.1/yolov5s.pt)

python3 train.py --img 640 --batch 16 --epochs 1 --data data/roadsign\_voc.yaml --cfg models/yolov5s.yaml --weights weights/yolov5s.pt --noautoanchor

基本上照着做就有了，但是有些参数要修改

参考文档：

[Hi3516上yolov5的pytorch转onnx转caffe转wk详解 - 盛夏夜 - 博客园](https://www.cnblogs.com/d442130165/p/16711189.html)