**【玩转YOLOv5】YOLOv5转openvino并进行部署**



[BIT可达鸭](https://blog.csdn.net/weixin_44936889) 2020-12-09 21:27:36  1447  收藏 15

分类专栏： [深度学习-目标检测](https://blog.csdn.net/weixin_44936889/category_9604789.html) 文章标签： [深度学习](https://www.csdn.net/tags/MtTaggxsNzMxNC1ibG9n.html) [目标检测](https://www.csdn.net/tags/MtTaEg0sMTMzODQtYmxvZwO0O0OO0O0O.html) [计算机视觉](https://www.csdn.net/tags/MtTaggwsNTg2MC1ibG9n.html) [pytorch](https://www.csdn.net/tags/MtTaEg0sMTI2NDMtYmxvZwO0O0OO0O0O.html) [openvion](https://so.csdn.net/so/search/s.do?q=openvion&t=blog&o=vip&s=&l=&f=&viparticle=)

版权

**【玩转YOLOv5】YOLOv5的Openvino转换和部署**

* [本文禁止转载！](https://blog.csdn.net/weixin_44936889/article/details/110940322#_1)
* [1. YOLOv5环境配置：](https://blog.csdn.net/weixin_44936889/article/details/110940322#1_YOLOv5_2)
* [2. 修改模型文件：](https://blog.csdn.net/weixin_44936889/article/details/110940322#2__6)
  + [1. models/yolo.py](https://blog.csdn.net/weixin_44936889/article/details/110940322#1_modelsyolopy_11)
  + [2. models/export.py](https://blog.csdn.net/weixin_44936889/article/details/110940322#2_modelsexportpy_24)
  + [3. utils/torch\_utils.py](https://blog.csdn.net/weixin_44936889/article/details/110940322#3_utilstorch_utilspy_37)
* [3. 训练模型：](https://blog.csdn.net/weixin_44936889/article/details/110940322#3__48)
* [4. torch模型转onnx：](https://blog.csdn.net/weixin_44936889/article/details/110940322#4_torchonnx_59)
* [5. onnx转openvino：](https://blog.csdn.net/weixin_44936889/article/details/110940322#5_onnxopenvino_88)
  + [5.1 激活环境：](https://blog.csdn.net/weixin_44936889/article/details/110940322#51__89)
  + [5.2 安装依赖：](https://blog.csdn.net/weixin_44936889/article/details/110940322#52__110)
  + [5.3 脚本转换：](https://blog.csdn.net/weixin_44936889/article/details/110940322#53__116)
* [6. 模型测试：](https://blog.csdn.net/weixin_44936889/article/details/110940322#6__137)
* [7. 交流群：](https://blog.csdn.net/weixin_44936889/article/details/110940322#7__388)
* [关注我的公众号：](https://blog.csdn.net/weixin_44936889/article/details/110940322#_391)

**本文禁止转载！**

**1. YOLOv5环境配置：**

可以看我之前写的几篇：  
[【小白CV教程】Pytorch训练YOLOv5并量化压缩（VOC格式数据集）](https://blog.csdn.net/weixin_44936889/article/details/110732476)

**2. 修改模型文件：**

由于YOLOv5的许多算子openvino仍然不支持，因此我们需要做出几点修改。首先我们需要将 Hardswish 激活函数替换掉，换成 LeackyReLU。

具体修改的地方有：

**1. models/yolo.py**

models/yolo.py：

# 第32行

self.act = nn.Hardswish() if act else nn.Identity()

* 1
* 2

修改为：

# 第32行

self.act = nn.LeakyReLU(0.1, inplace=True) if act else nn.Identity()

* 1
* 2

**2. models/export.py**

# 第46行

if isinstance(m, models.common.Conv) and isinstance(m.act, nn.Hardswish):

m.act = Hardswish() # assign activation

* 1
* 2
* 3

修改为：

# 第46行

if isinstance(m, models.common.Conv) and isinstance(m.act, nn.LeakyReLU):

m.act = LeakyReLU() # assign activation

* 1
* 2
* 3

**3. utils/torch\_utils.py**

# 第90行

elif t in [nn.Hardswish, nn.LeakyReLU, nn.ReLU, nn.ReLU6]:

* 1
* 2

修改为：

# 第90行

elif t in [nn.LeakyReLU, nn.LeakyReLU, nn.ReLU, nn.ReLU6]:

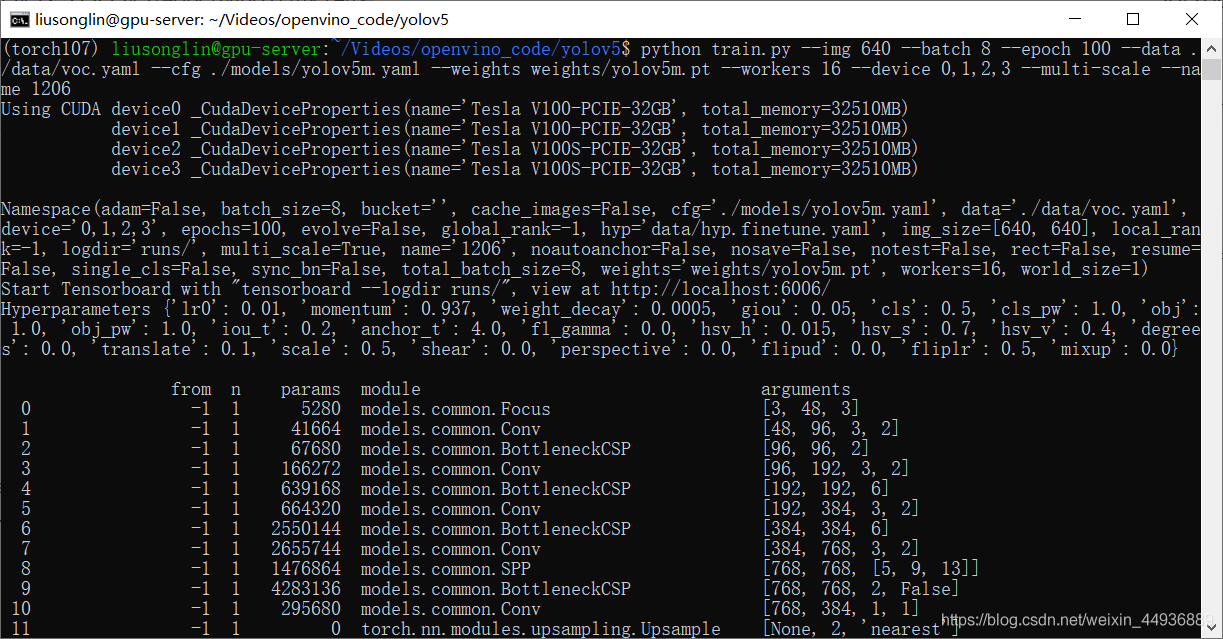
* 1
* 2

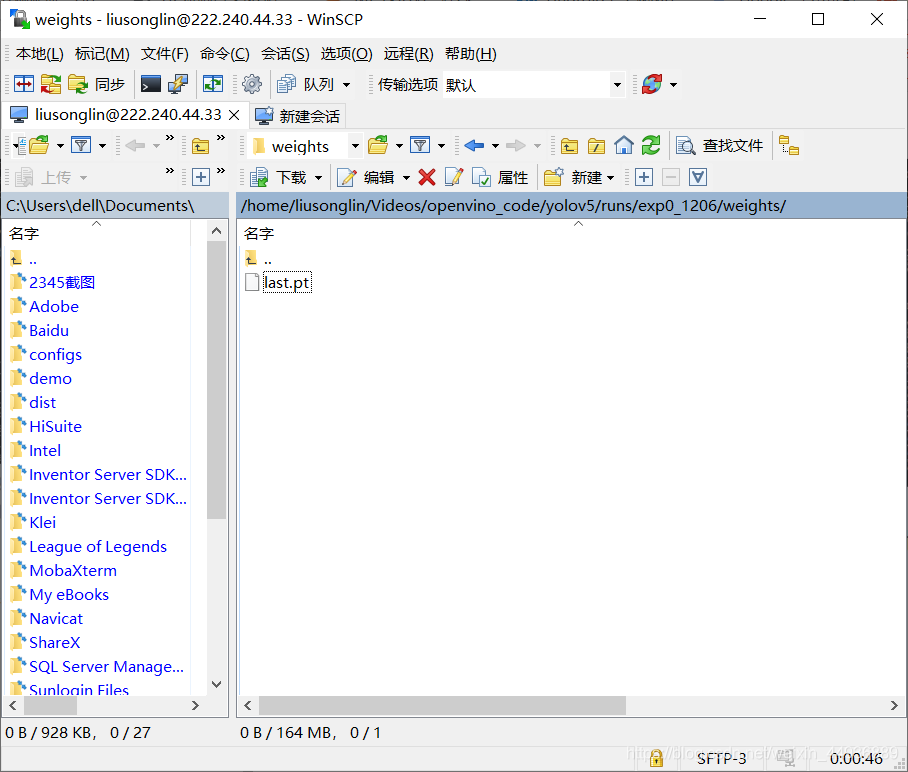
**3. 训练模型：**

然后训练模型：

python train.py --img 640 --batch 8 --epoch 100 --data ./data/voc.yaml --cfg ./models/yolov5m.yaml --weights weights/yolov5m.pt --workers 16 --device 0,1,2,3 --multi-scale --name 1206

* 1

  
其中 name 参数是模型保存的路径，我的训练好的模型权重就被保存在 runs/train/exp0\_1206/weights 文件夹下，名称为 last.pt：



**4. torch模型转onnx：**

yolov5 官方给出了转换的代码，就是 models/export.py 代码，由于我们还需要将他转到 openvino，所以我们需要做出一点修改：

# 第53行

torch.onnx.export(model, img, f, verbose=False, opset\_version=11, input\_names=['data'],

output\_names=['prob']if y is None else ['output'])

* 1
* 2
* 3

修改为：

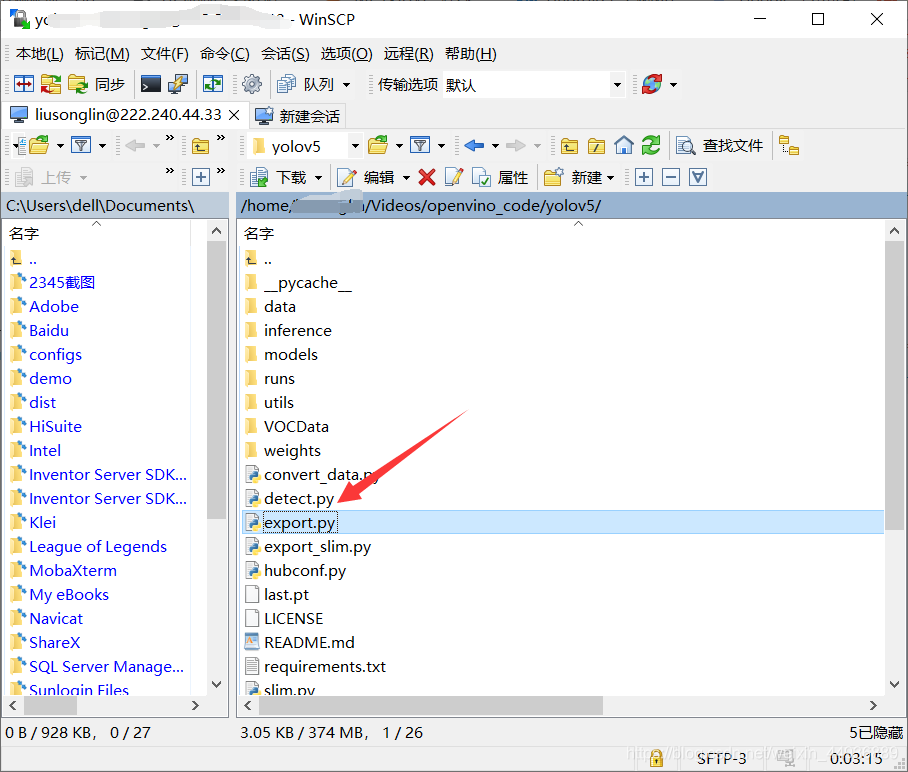
# 第53行

torch.onnx.export(model, img, f, verbose=False, opset\_version=10, input\_names=['data'],

output\_names=['prob']if y is None else ['output'])

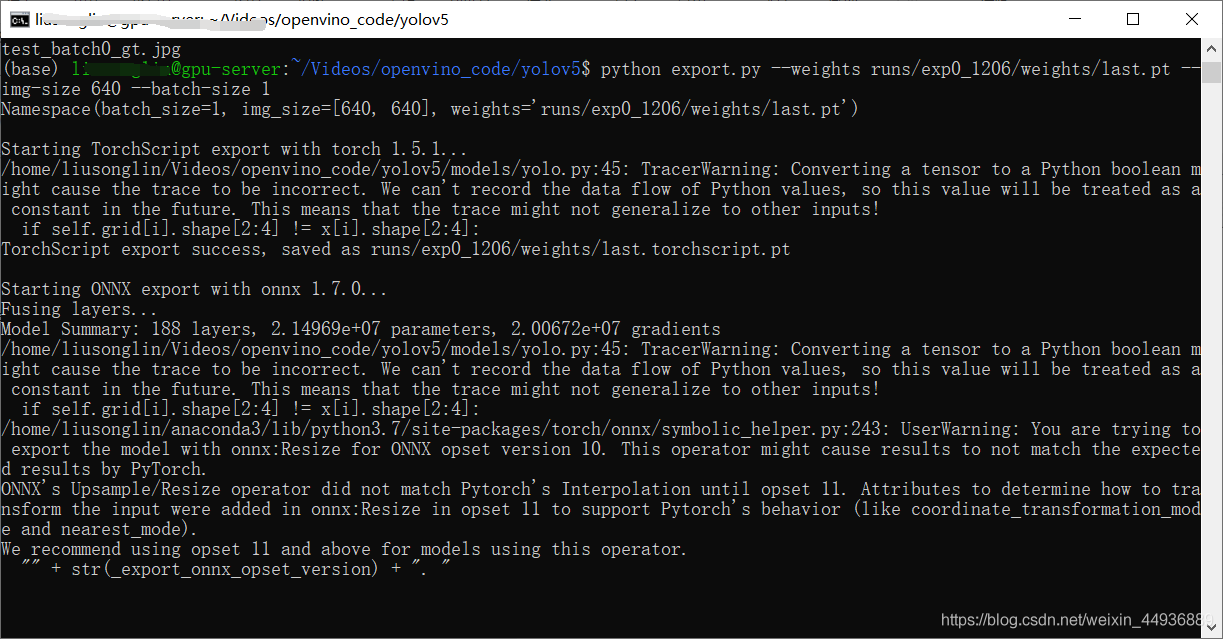
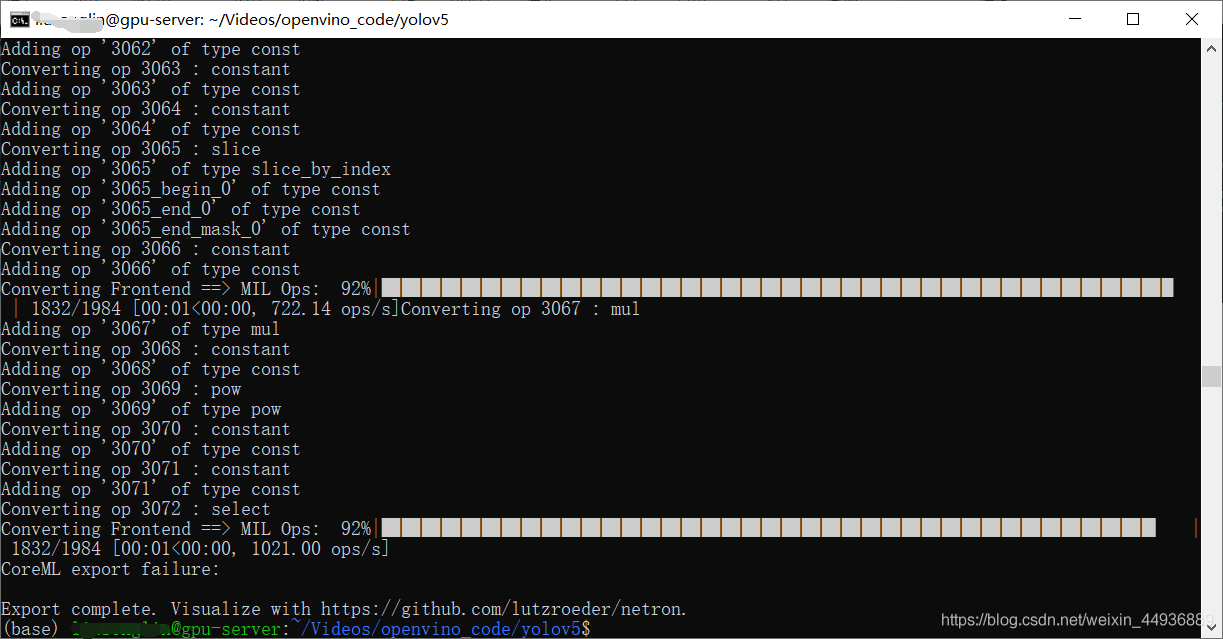
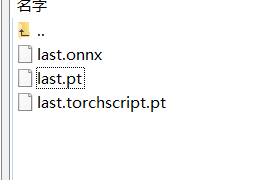
* 1
* 2
* 3

否则 11 版本的算子库在转到openvino会报错。

将 export.py 放置在根目录：  
  
运行以下命令进行转换：

python export.py --weights runs/exp0\_1206/weights/last.pt --img-size 640 --batch-size 1

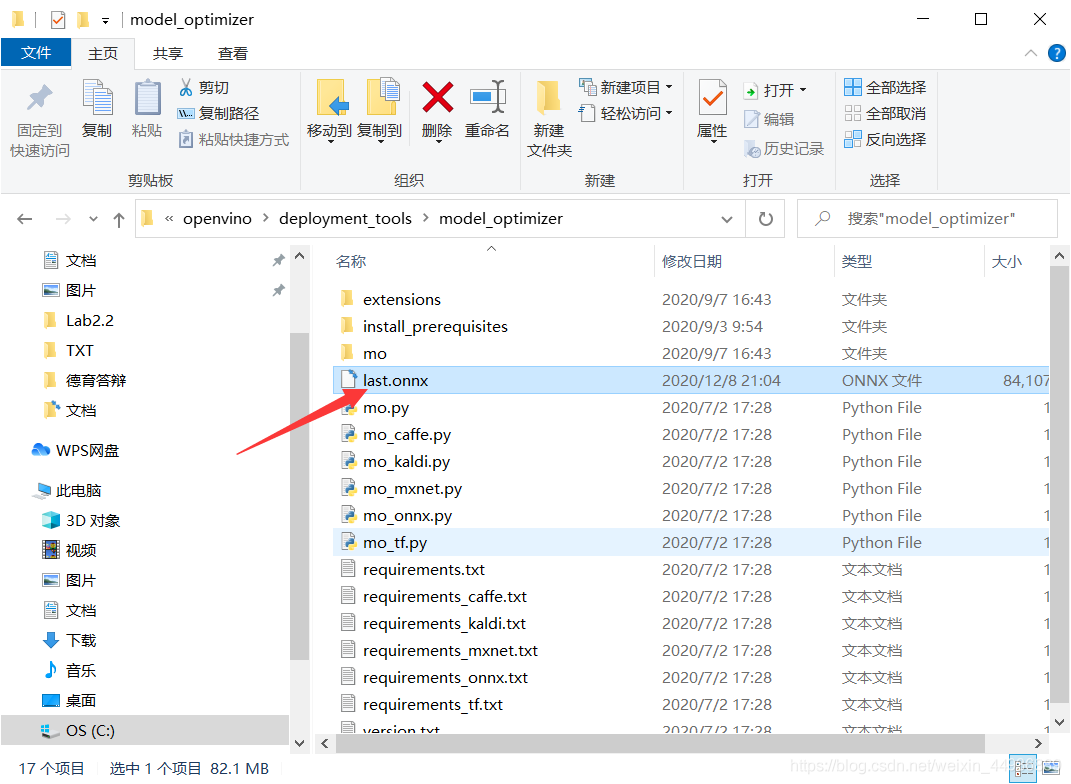
* 1

  
转换成功：  
  
可以看到 runs/exp0\_1206/weights/ 目录下生成了 last.onnx 文件：  


**5. onnx转openvino：**

**5.1 激活环境：**

这里我没有在 Linux 服务器配置 openvino 环境，因此我下载到我的 Windows 进行转换。

首先我们打开 openvino 模型转换目录，将 last.onnx 拷贝进去：  
  
我本地的目录是：

C:\Program Files (x86)\IntelSWTools\openvino\deployment\_tools\model\_optimizer

* 1

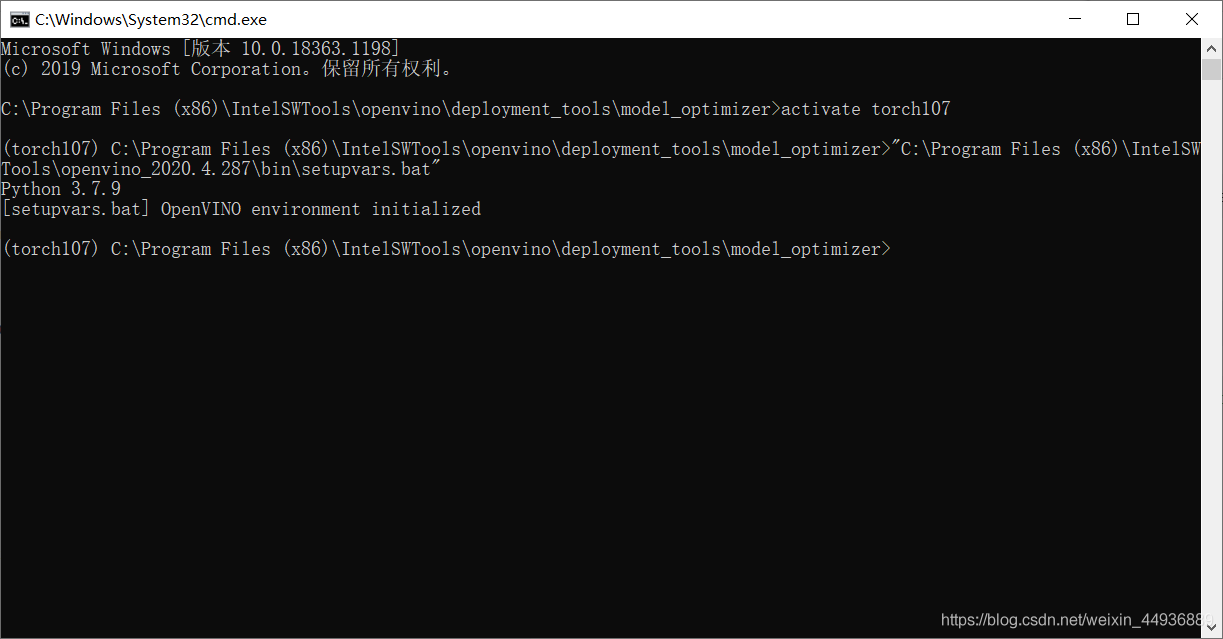
在该目录打开 cmd，激活 conda 和 openvino环境：

activate torch107

* 1

"C:\Program Files (x86)\IntelSWTools\openvino\_2020.4.287\bin\setupvars.bat"

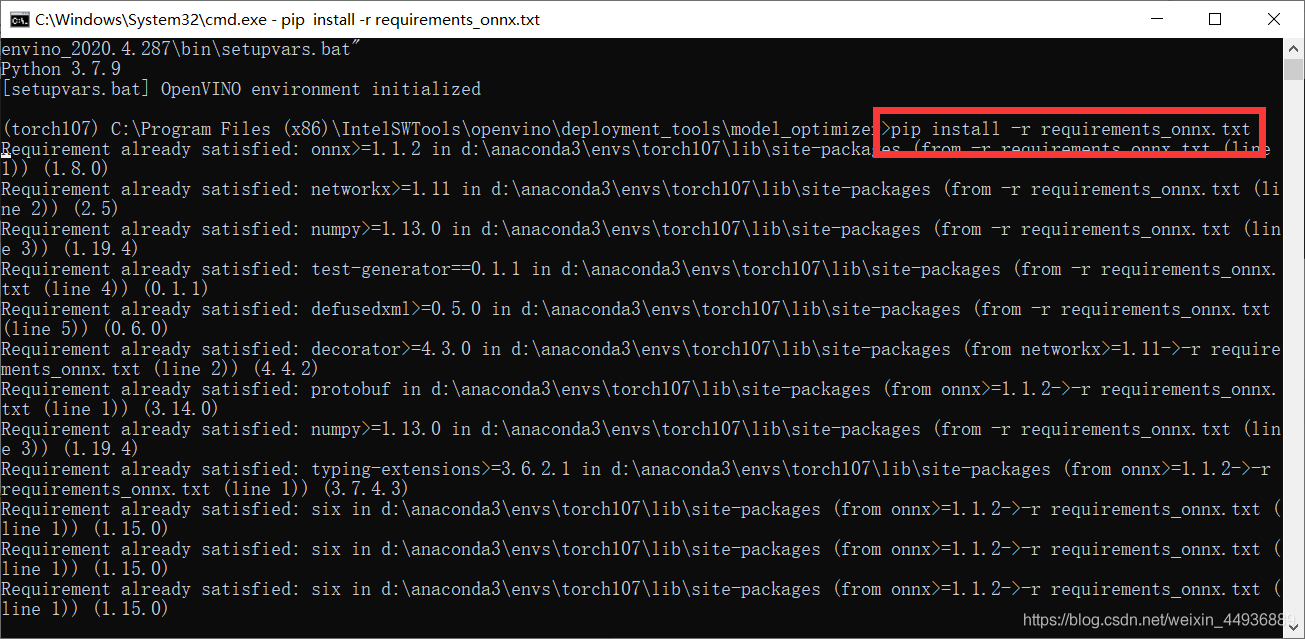
* 1



**5.2 安装依赖：**

pip install -r requirements\_onnx.txt

* 1

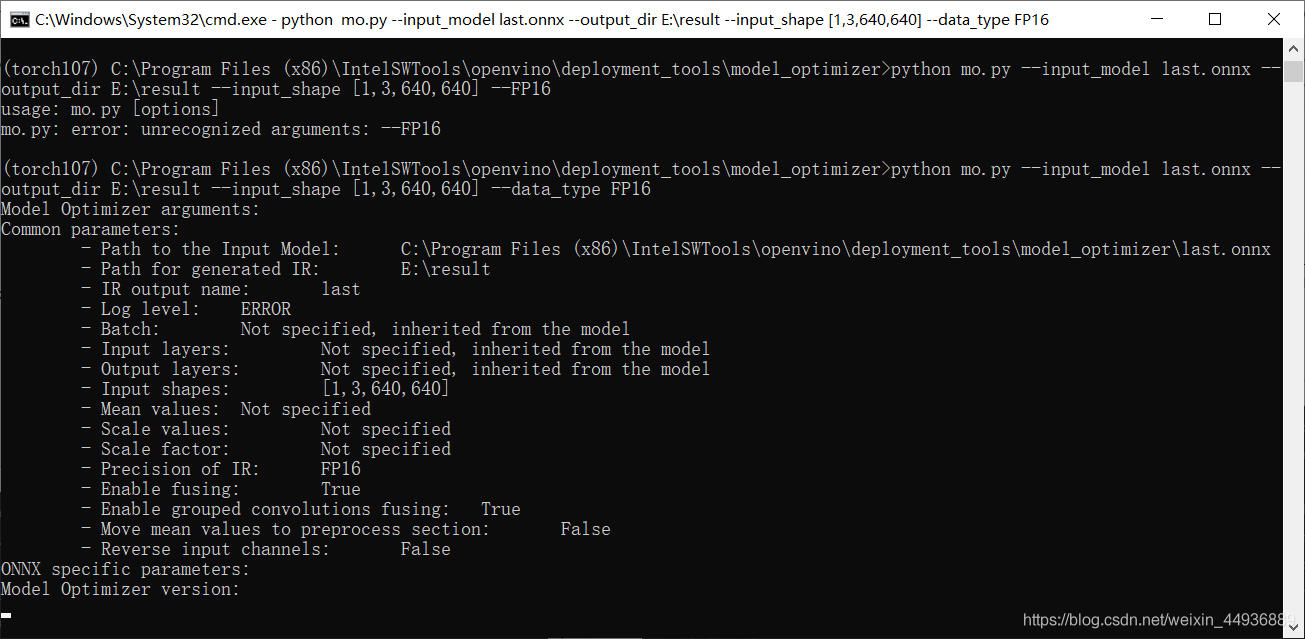


**5.3 脚本转换：**

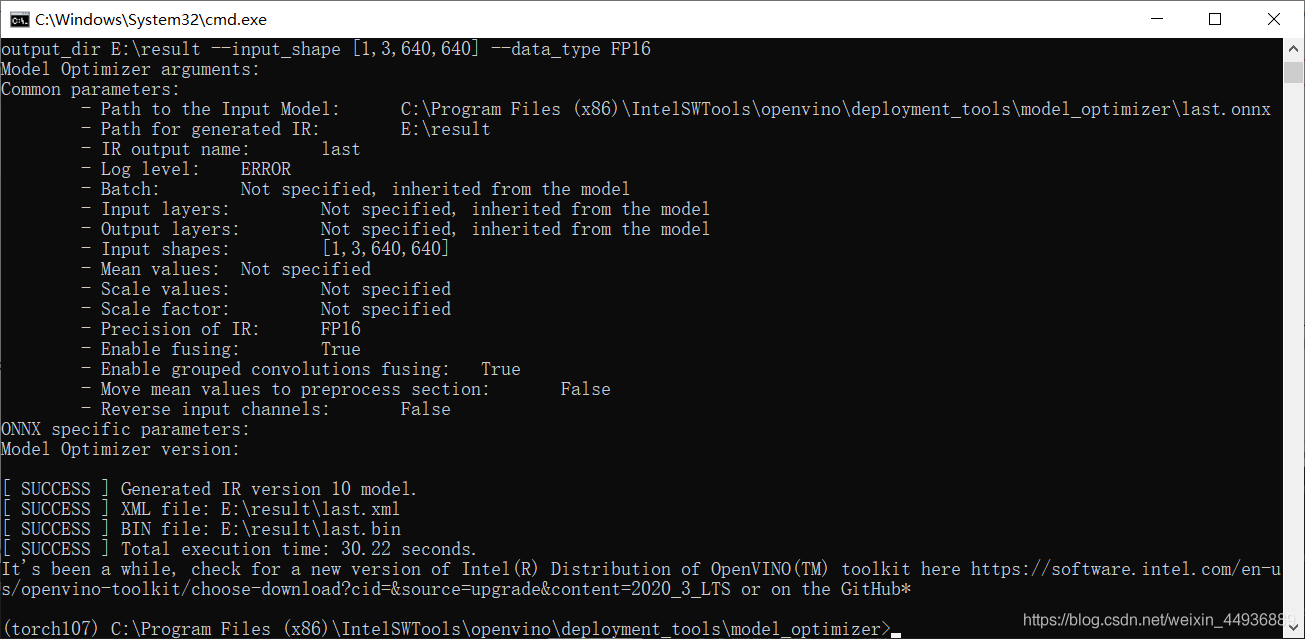
运行模型转换脚本：

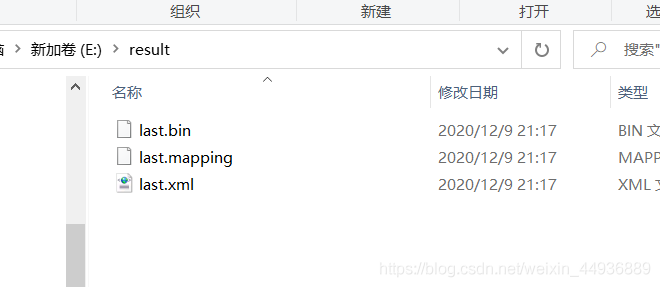
python mo.py --input\_model last.onnx --output\_dir E:\result --input\_shape [1,3,640,640] --data\_type FP16

* 1

（这里导出半精度模型）  


其他转换参数可以查看：  
<https://docs.openvinotoolkit.org/cn/latest/_docs_MO_DG_prepare_model_convert_model_Converting_Model_General.html>  


转换成功：  
  
可以看到模型输出路径下生成了 bin 和 xml 文件：



**6. 模型测试：**

我们创建 run.py：

from \_\_future\_\_ import print\_function

import logging as log

import os

import pathlib

import json

import cv2

import numpy as np

from openvino.inference\_engine import IENetwork, IECore

import torch

import torchvision

import time

def xywh2xyxy(x):

# Convert nx4 boxes from [x, y, w, h] to [x1, y1, x2, y2] where xy1=top-left, xy2=bottom-right

y = torch.zeros\_like(x) if isinstance(

x, torch.Tensor) else np.zeros\_like(x)

y[:, 0] = x[:, 0] - x[:, 2] / 2 # top left x

y[:, 1] = x[:, 1] - x[:, 3] / 2 # top left y

y[:, 2] = x[:, 0] + x[:, 2] / 2 # bottom right x

y[:, 3] = x[:, 1] + x[:, 3] / 2 # bottom right y

return y

def non\_max\_suppression(prediction, conf\_thres=0.1, iou\_thres=0.6, merge=False, classes=None, agnostic=False):

"""Performs Non-Maximum Suppression (NMS) on inference results

Returns:

detections with shape: nx6 (x1, y1, x2, y2, conf, cls)

"""

prediction = torch.from\_numpy(prediction)

if prediction.dtype is torch.float16:

prediction = prediction.float() # to FP32

nc = prediction[0].shape[1] - 5 # number of classes

xc = prediction[..., 4] > conf\_thres # candidates

# Settings

# (pixels) minimum and maximum box width and height

min\_wh, max\_wh = 2, 4096

max\_det = 300 # maximum number of detections per image

time\_limit = 10.0 # seconds to quit after

redundant = True # require redundant detections

multi\_label = nc > 1 # multiple labels per box (adds 0.5ms/img)

t = time.time()

output = [None] \* prediction.shape[0]

for xi, x in enumerate(prediction): # image index, image inference

# Apply constraints

# x[((x[..., 2:4] < min\_wh) | (x[..., 2:4] > max\_wh)).any(1), 4] = 0 # width-height

x = x[xc[xi]] # confidence

# If none remain process next image

if not x.shape[0]:

continue

# Compute conf

x[:, 5:] \*= x[:, 4:5] # conf = obj\_conf \* cls\_conf

# Box (center x, center y, width, height) to (x1, y1, x2, y2)

box = xywh2xyxy(x[:, :4])

# Detections matrix nx6 (xyxy, conf, cls)

if multi\_label:

i, j = (x[:, 5:] > conf\_thres).nonzero(as\_tuple=False).T

x = torch.cat((box[i], x[i, j + 5, None], j[:, None].float()), 1)

else: # best class only

conf, j = x[:, 5:].max(1, keepdim=True)

x = torch.cat((box, conf, j.float()), 1)[

conf.view(-1) > conf\_thres]

# Filter by class

if classes:

x = x[(x[:, 5:6] == torch.tensor(classes, device=x.device)).any(1)]

# Apply finite constraint

# if not torch.isfinite(x).all():

# x = x[torch.isfinite(x).all(1)]

# If none remain process next image

n = x.shape[0] # number of boxes

if not n:

continue

# Sort by confidence

# x = x[x[:, 4].argsort(descending=True)]

# Batched NMS

c = x[:, 5:6] \* (0 if agnostic else max\_wh) # classes

# boxes (offset by class), scores

boxes, scores = x[:, :4] + c, x[:, 4]

i = torchvision.ops.boxes.nms(boxes, scores, iou\_thres)

if i.shape[0] > max\_det: # limit detections

i = i[:max\_det]

if merge and (1 < n < 3E3): # Merge NMS (boxes merged using weighted mean)

try: # update boxes as boxes(i,4) = weights(i,n) \* boxes(n,4)

iou = box\_iou(boxes[i], boxes) > iou\_thres # iou matrix

weights = iou \* scores[None] # box weights

x[i, :4] = torch.mm(weights, x[:, :4]).float(

) / weights.sum(1, keepdim=True) # merged boxes

if redundant:

i = i[iou.sum(1) > 1] # require redundancy

except: # possible CUDA error https://github.com/ultralytics/yolov3/issues/1139

print(x, i, x.shape, i.shape)

pass

output[xi] = x[i]

if (time.time() - t) > time\_limit:

break # time limit exceeded

return output

device = 'CPU'

# device = 'CPU'

input\_h, input\_w, input\_c, input\_n = (640, 640, 3, 1)

log.basicConfig(level=log.DEBUG)

# For objection detection task, replace your target labels here.

label\_id\_map = ["face", "normal", "phone",

"write", "smoke", "eat", "computer", "sleep"]

exec\_net = None

def init(model\_xml):

if not os.path.isfile(model\_xml):

log.error(f'{model\_xml} does not exist')

return None

model\_bin = pathlib.Path(model\_xml).with\_suffix('.bin').as\_posix()

net = IENetwork(model=model\_xml, weights=model\_bin)

ie = IECore()

global exec\_net

exec\_net = ie.load\_network(network=net, device\_name=device)

input\_blob = next(iter(net.inputs))

n, c, h, w = net.inputs[input\_blob].shape

global input\_h, input\_w, input\_c, input\_n

input\_h, input\_w, input\_c, input\_n = h, w, c, n

return net

def process\_image(net, input\_image):

if not net or input\_image is None:

log.error('Invalid input args')

return None

ih, iw, \_ = input\_image.shape

if ih != input\_h or iw != input\_w:

input\_image = cv2.resize(input\_image, (input\_w, input\_h))

input\_image = cv2.cvtColor(input\_image, cv2.COLOR\_BGR2RGB)

input\_image = input\_image/255

input\_image = input\_image.transpose((2, 0, 1))

images = np.ndarray(shape=(input\_n, input\_c, input\_h, input\_w))

images[0] = input\_image

input\_blob = next(iter(net.inputs))

out\_blob = next(iter(net.outputs))

start = time.time()

res = exec\_net.infer(inputs={input\_blob: images})

end = time.time()

print('-[INFO] inference time: {}ms'.format(end - start))

data = res[out\_blob]

data = non\_max\_suppression(data, 0.4, 0.5)

detect\_objs = []

if data[0] == None:

return json.dumps({"objects": detect\_objs})

else:

data = data[0].numpy()

for proposal in data:

if proposal[4] > 0:

confidence = proposal[4]

xmin = np.int(iw \* (proposal[0]/640))

ymin = np.int(ih \* (proposal[1]/640))

xmax = np.int(iw \* (proposal[2]/640))

ymax = np.int(ih \* (proposal[3]/640))

detect\_objs.append((

int(xmin),

int(ymin),

int(xmax),

int(ymax),

label\_id\_map[int(proposal[5])],

float(confidence)

))

return detect\_objs

def plot\_bboxes(image, bboxes, line\_thickness=None):

# Plots one bounding box on image img

tl = line\_thickness or round(

0.002 \* (image.shape[0] + image.shape[1]) / 2) + 1 # line/font thickness

for (x1, y1, x2, y2, cls\_id, pos\_id) in bboxes:

if cls\_id == 'smoke' or cls\_id == 'phone':

color = (0, 0, 255)

else:

color = (0, 255, 0)

c1, c2 = (x1, y1), (x2, y2)

cv2.rectangle(image, c1, c2, color, thickness=tl, lineType=cv2.LINE\_AA)

tf = max(tl - 1, 1) # font thickness

t\_size = cv2.getTextSize(cls\_id, 0, fontScale=tl / 3, thickness=tf)[0]

c2 = c1[0] + t\_size[0], c1[1] - t\_size[1] - 3

cv2.rectangle(image, c1, c2, color, -1, cv2.LINE\_AA) # filled

cv2.putText(image, '{} ID-{}'.format(cls\_id, pos\_id), (c1[0], c1[1] - 2), 0, tl / 3,

[225, 255, 255], thickness=tf, lineType=cv2.LINE\_AA)

return image

if \_\_name\_\_ == '\_\_main\_\_':

# Test API

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument('--model\_xml', type=str, default='result\last.xml')

parser.add\_argument('--source', type=str, default='images')

opt = parser.parse\_args()

predictor = init(opt.model\_xml)

for p in os.listdir(opt.source):

img = cv2.imread(os.path.join(opt.source, p))

result = process\_image(predictor, img)

img = plot\_bboxes(img, result)

cv2.imshow('result', img)

cv2.waitKey(0)

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82
* 83
* 84
* 85
* 86
* 87
* 88
* 89
* 90
* 91
* 92
* 93
* 94
* 95
* 96
* 97
* 98
* 99
* 100
* 101
* 102
* 103
* 104
* 105
* 106
* 107
* 108
* 109
* 110
* 111
* 112
* 113
* 114
* 115
* 116
* 117
* 118
* 119
* 120
* 121
* 122
* 123
* 124
* 125
* 126
* 127
* 128
* 129
* 130
* 131
* 132
* 133
* 134
* 135
* 136
* 137
* 138
* 139
* 140
* 141
* 142
* 143
* 144
* 145
* 146
* 147
* 148
* 149
* 150
* 151
* 152
* 153
* 154
* 155
* 156
* 157
* 158
* 159
* 160
* 161
* 162
* 163
* 164
* 165
* 166
* 167
* 168
* 169
* 170
* 171
* 172
* 173
* 174
* 175
* 176
* 177
* 178
* 179
* 180
* 181
* 182
* 183
* 184
* 185
* 186
* 187
* 188
* 189
* 190
* 191
* 192
* 193
* 194
* 195
* 196
* 197
* 198
* 199
* 200
* 201
* 202
* 203
* 204
* 205
* 206
* 207
* 208
* 209
* 210
* 211
* 212
* 213
* 214
* 215
* 216
* 217
* 218
* 219
* 220
* 221
* 222
* 223
* 224
* 225
* 226
* 227
* 228
* 229
* 230

运行：

python --model\_xml result\last.xml --source images

* 1

其中两个参数分别为模型路径和测试图片路径，运行结果如图：  


大概 CPU 能跑到 200+ FPS。