实用教程详解：用OpenCV的DNN模块部署YOLOv5目标检测

原创 CV开发者都爱看的 极市平台 2021-01-19 18:15:00 手机阅读

作者丨nihate

审稿丨邓富城

编辑丨极市平台

极市导读

本文中介绍的整套程序只依赖OpenCV库就能正常运行，彻底摆脱了对深度学习框架的依赖。文章讲述了作者在自己编写用OpenCV的dnn模块做YOLOv5目标检测的程序的过程中遇到的bug以及解决的办法。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

最近看到多篇讲解YOLOv5在OpenVINO部署做目标检测文章，但是没看到过用OpenCV的DNN模块做YOLOv5目标检测的。于是，我就想编写一套用OpenCV的DNN模块做YOLOv5目标检测的程序。

在编写这套程序时，遇到的bug和解决办法，在这篇文章里讲述一下。

在YOLOv5之前的YOLOv3和YOLOv4的官方代码都是基于darknet框架的实现的，因此OpenCV的DNN模块做目标检测时，读取的是.cfg和.weight文件，那时候编写程序很顺畅，没有遇到bug。

但是YOLOv5的官方代码(https://github.com/ultralytics/yolov5)是基于Pytorch框架实现的，而OpenCV的DNN模块不支持读取Pytorch的训练模型文件。如果想要把Pytorch的训练模型.pth文件加载到OpenCV的DNN模块里，需要先把Pytorch的训练模型.pth文件转换到.onnx文件，然后才能载入到Opencv的DNN模块里。

因此，用OpenCV的DNN模块做YOLOv5目标检测的程序，包含两个步骤：

1. 把Pytorch的训练模型.pth文件转换到.onnx文件。

2. OpenCV的DNN模块读取.onnx文件做前向计算。

1. 把Pytorch的训练模型.pth文件转换到.onnx文件

在做这一步时，我得吐槽一下官方代码：

https://github.com/ultralytics/yolov5

这套程序里的代码混乱，在Pytorch里，通常是在.py文件里定义网络结构的，但是官方代码是在.yaml文件定义网络结构，利用Pytorch动态图特性，解析.yaml文件自动生成网络结构。在.yaml文件里有depth_multiple和width_multiple，它是控制网络的深度和宽度的参数。

这么做的好处是能够灵活的配置网络结构，但是不利于理解网络结构。假如你想设断点查看某一层的参数和输出数值，那就没办法了。因此，在我编写的转换到.onnx文件的程序里，网络结构是在.py文件里定义的。

其次，在官方代码里，还有一个奇葩的地方，那就是.pth文件。

起初，我下载官方代码到本地运行时，torch.load读取.pth文件总是出错，后来把pytorch升级到1.7，就读取成功了。可以看到版本兼容性不好，这是它的一个不足之处。

设断点查看读取的.pth文件里的内容，可以看到.pth里既存储有模型参数，也存储有网络结构，还储存了一些超参数，包括anchors，stride等等的。第一次见到有这种操作的，通常情况下，.pth文件里只存储了训练模型参数的。

查看models\yolo.py里的Detect类，在构造函数里，有这么两行代码：

```
self.register_buffer('anchors', a)  # shape(nl,na,2)
self.register_buffer('anchor_grid', a.clone().view(self.nl, 1, -1, 1, 1, 2))
```

我尝试过把这两行代码改成self.anchors = a 和 self.anchor_grid = a.clone().view(self.nl, 1, -1, 1, 1, 2)，程序依然能正常运行，但是torch.save保存模型文件后，可以看到.pth文件里没有存储anchors和anchor_grid了，在网页搜索register_buffer，解释是：pytorch中register_buffer模型保存和加载的时候可以写入和读出。

在这两行代码的下一行：

```
self.m = nn.ModuleList(nn.Conv2d(x, self.no * self.na, 1) for x in ch)  # output conv
```

它的作用是做特征图的输出通道对齐，通过1x1卷积把三种尺度特征图的输出通道都调整到 num_anchors*(num_classes+5)。

阅读Detect类的forward函数代码，可以看出它的作用是根据偏移公式计算出预测框的中心坐标和高宽，这里需要注意的是，计算高和宽的代码：

pwh = (ps[:, 2:4].sigmoid() * 2) ** 2 * anchors[i]

没有采用exp操作，而是直接乘上anchors[i]，这是YOLOv5与YOLOv3v4的一个最大区别（还有一个区别就是在训练阶段的loss函数里，YOLOv5采用邻域的正样本anchor匹配策略，增加了正样本。其它的是一些小区别，比如YOLOv5的第一个模块采用FOCUS把输入数据2倍下采样切分成4份，在channel维度进行拼接，然后进行卷积操作，YOLOv5的激活函数没有使用Mish）。

现在可以明白Detect类的作用是计算预测框的中心坐标和高宽，简单来说就是生成proposal，作为后续NMS的输入，进而输出最终的检测框。我觉得在Detect类里定义的1x1卷积是不恰当的，应该把它定义在Detect类的外面，紧邻着Detect类之前定义1x1卷积。

在官方代码里，有转换到onnx文件的程序：python models/export.py --weights yolov5s.pt --img 640 --batch 1

在pytorch1.7版本里，程序是能正常运行生成onnx文件的。观察export.py里的代码，在执行torch.onnx.export之前，有这么一段代码：

```
# Input
img = torch.zeros(opt.batch_size, 3, *opt.img_size)  # image size(1,3,320,192) iDetection

# Update model
for k, m in model.named_modules():
    m._non_persistent_buffers_set = set()  # pytorch 1.6.0 compatibility
    if isinstance(m, models.common.Conv):  # assign export-friendly activations
        if isinstance(m.act, nn.Hardswish):
            m.act = Hardswish()
        elif isinstance(m.act, nn.SiLU):
            m.act = SiLU()
    # elif isinstance(m, models.yolo.Detect):
    #     m.forward = m.forward_export  # assign forward (optional)
model.model[-1].export = True  # set Detect() layer export=True
y = model(img)  # dry run
```


注意其中的for循环，我试验过注释掉它，重新运行就会出错，打印出的错误如下：

```
Starting ONNX export with onnx 1.8.0...
ONNX export failure: Exporting the operator silu to ONNX opset version 12 is not supported. Please open a bug to request ONNX export support for the missing operator.
```

由此可见，这段for循环代码是必需的。

2. OpenCV的DNN模块读取.onnx文件做前向计算

在生成.onnx文件后，就可以用OpenCV的DNN模块里的cv2.dnn.readNet读取它。然而，在读取时，出现了如下错误：

```
    net = cv2.dnn.readNet(output_onnx)
cv2.error: OpenCV(4.4.0) /tmp/pip-req-build-qacpj5ci/opencv/modules/dnn/src/onnx/onnx_importer.cpp:562: error: (-2:Unspecified error) in function 'void cv::d
> Slice layer only supports steps = 1 (expected: 'countNonZero(step_blob != 1) == 0'), where
>     'countNonZero(step_blob != 1)' is 1
> must be equal to
>     '0' is 0
```

我在网页搜索这个问题的解决办法，看到一篇技术文章(https://zhuanlan.zhihu.com/p/286298001)，文章里讲述的第一条：

（1）Pytorch2ONNX不支持对slice对象赋值

下面这段代码是不被Pytorch原生的onnx转换接口支持的，即不能对slice对象赋值

```
preds[:, :, y1:y2, x1:x2] += crop_seg_logit
```


于是查看YOLOv5的代码，在common.py文件的Focus类，torch.cat的输入里有4次切片操作，代码如下：

```
class Focus(nn.Module):
    # Focus wh information into c-space
    def __init__(self, c1, c2, k=1, s=1, p=None, g=1, act=True):  # ch_in, ch_out, kernel, stride, padding, groups
        super(Focus, self).__init__()
        self.conv = Conv(c1 * 4, c2, k, s, p, g, act)
        # self.contract = Contract(gain=2)

    def forward(self, x):  # x(b,c,w,h) -> y(b,4c,w/2,h/2)
        return self.conv(torch.cat([x[..., ::2, ::2], x[..., 1::2, ::2], x[..., ::2, 1::2], x[..., 1::2, 1::2]], 1))
        # return self.conv(self.contract(x))
```

那么现在需要更换索引式的切片操作，观察到注释的Contract类，它就是用view和permute函数完成切片操作的，于是修改代码如下：

```
class Focus(nn.Module):
    # Focus wh information into c-space
    def __init__(self, c1, c2, k=1, s=1, p=None, g=1, act=True):  # ch_in, ch_out, kernel, stride, padding, grou
        super(Focus, self).__init__()
        self.conv = Conv(c1 * 4, c2, k, s, p, g, act)
        self.contract = Contract(gain=2)

    def forward(self, x):  # x(b,c,w,h) -> y(b,4c,w/2,h/2)
        # return self.conv(torch.cat([x[..., ::2, ::2], x[..., 1::2, ::2], x[..., ::2, 1::2], x[..., 1::2, 1::2]
        return self.conv(self.contract(x))

class Contract(nn.Module):
    # Contract width-height into channels, i.e. x(1,64,80,80) to x(1,256,40,40)
    def __init__(self, gain=2):
        super().__init__()
        self.gain = gain
```


其次，在models\yolo.py里的Detect类里，也有切片操作，代码如下：

```
y[..., 0:2] = (y[..., 0:2] * 2. - 0.5 + self.grid[i].to(x[i].device)) * self.stride[i]
y[..., 2:4] = (y[..., 2:4] * 2) ** 2 * self.anchor_grid[i]  # wh
```

前面说过，Detect类的作用是计算预测框的中心坐标和高宽，生成proposal，这个是属于后处理的，因此不需要把它写入到onnx文件里。

总结一下，按照上面的截图代码，修改Focus类，把Detect类里面的1x1卷积定义在紧邻着Detect类之前的外面，然后去掉Detect类，组成新的model，作为torch.onnx.export的输入，

torch.onnx.export(model, inputs, output_onnx, verbose=False, opset_version=12, input_names=['images'], output_names=['out0', 'out1', 'out2'])

最后生成的onnx文件，opencv的dnn模块就能成功读取了，接下来对照Detect类里的forward函数，用python或者C++编写计算预测框的中心坐标和高宽的功能。

周末这两天，我在win10+cpu机器里编写了用OpenCV的DNN模块做Yolov5目标检测的程序，包含Python和C++两个版本的。程序都调试通过了，运行结果也是正确的。

我把这套代码发布在了Github上，地址是：

https://github.com/hpc203/yolov5-dnn-cpp-python

后处理模块，python版本用numpy array实现的，C++版本的用vector和数组实现的，整套程序只依赖OpenCV库(opencv4版本以上的)就能正常运行，彻底摆脱对深度学习框架pytorch,tensorflow,caffe,mxnet等等的依赖。

用OpenVINO作目标检测，需要把onnx文件转换到.bin和.xml文件，相比于用DNN模块加载onnx文件做目标检测是多了一个步骤的。因此，我就想编写一套用OpenCV的DNN模块做YOLOv5目标检测的程序，用Opencv的DNN模块做深度学习目标检测，在win10和ubuntu，在cpu和gpu上都能运行，可见DNN模块的通用性更好，很接地气。

◎作者档案

作者：nihate
欢迎大家联系极市小编（微信ID:fengcall19）加入极市原创作者行列

推荐阅读

- 完整教程：使用YOLO V5训练自动驾驶目标检测网络
- 工程案例详解：YOLOv5在建筑工地中安全帽佩戴检测的应用
- YOLO窥见黑夜丨YOLO in the Dark让黑夜里的目标检测成为可能

阅读原文

喜欢此内容的人还喜欢

15个目标检测开源数据集汇总

极市平台