# onnx 结构优化

***author: qiuchao***

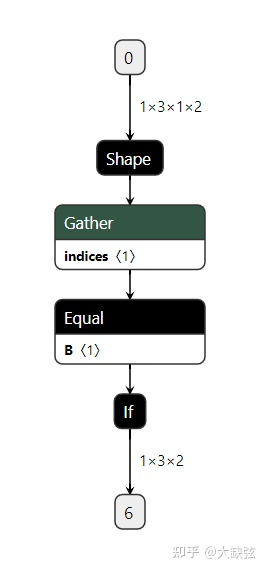
# 一，介绍

常用onnx simplifier（简称 onnxsim） 的小伙伴可能知道，onnxsim 本身只提供 constant folding/propagation（即消除结果恒为常量的算子）的能力，而图变换（即合并 conv 和 bn 等等）的能力是由 onnxsim 调用 onnx optimizer 的各种 pass 实现的。constant folding 和图变换同时使用时，很多隐藏的优化机会会被挖掘出来，这也是 onnxsim 优化效果出色的原因之一。例如 add(add(x, 1), 2) 在变换为 add(x, add(1, 2)) 之后就可以通过 constant folding 变为 add(x, 3)，而 pad(conv(x, w, padding=0), add(1, 1)) 在经过 constant folding 变为 pad(conv(x, w, padding=0), 2) 后，就可以进一步融合成 conv(x, w, padding=2)。[点此进入project](https://github.com/daquexian/onnx-simplifier)

借助最新版的 onnx optimizer，onnxsim 可以完美的优化 PyTorch squeeze op 带来的冗余操作。具体来说，一段只包含 squeeze 操作的 PyTorch 代码：

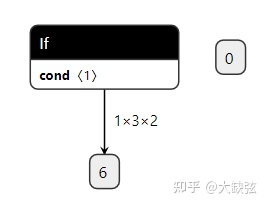
class Net(nn.Module):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 def forward(self, x):  
 return torch.squeeze(x, dim=2)  
  
net = Net()  
torch.onnx.export(net, torch.ones(1,3,1,2), 'squeeze.onnx', opset\_version=11)

导出的 onnx 模型如下图（netron 没有把模型结构显示完全，If node 里还包含了 true 和 false 两个未显示的子图，true 子图是一个 squeeze op，false 子图是一个 identity op）：



这个 onnx 模型这么复杂的原因是因为 onnx 的 squeeze op 和 pytorch squeeze op 的行为不完全一样：当 squeezed dim 那一维的长度不为 1 的时候 onnx squeeze op 会抛出错误，而 pytorch 则会让输出等于输入。这个复杂的 onnx 模型就是想和 pytorch 的行为对齐：先通过 Shape、Gather op 得到 dim 那一维的长度，再通过 Equal 和 If op 判断长度是不是 1，如果是 1 才运行 Squeeze op，否则运行 Identity op。

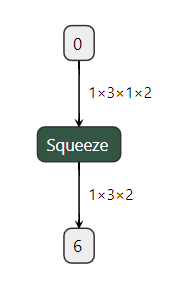
但是实际上对这个模型来说，这一大堆 op 都是没有必要的，因为输入形状是已知的 (1, 3, 1, 2)，squeeze dim 的长度是 1，所以一个普通的 onnx squeeze op 就足够了。onnxsim 会先用 constant folding 优化掉 Shape、Gather、Equal op，变成下面的样子（图片里 "0" 游离在外面也是因为 netron 的显示问题，它实际上被 If node 里的子图使用，netron 没有显示出来）：



此外 If node 的输入 cond 此时也已经是一个恒为 true 的常数。



也就是说 If 一定会走到 true 这个分支。接着 onnxsim 会调用新版 onnx optimizer 里刚刚实现的消除死分支的 pass，把这个模型里的 If op 删掉，把 true 分支提取出来，让这个复杂的 onnx 模型变成了它该有的样子：



# 二，使用方法

import onnx  
import onnxsim  
  
def simplifyONNX(od):  
 model\_onnx = onnx.load(od) # load onnx model  
 onnx.checker.check\_model(model\_onnx) # check onnx model  
 try:  
 ¦ print(f'simplifying with onnx-simplifier {onnxsim.\_\_version\_\_}...')  
 ¦ model\_onnx, check = onnxsim.simplify(model\_onnx)  
 ¦ assert check, 'assert check failed'  
 ¦ onnx.save(model\_onnx,od)  
 except Exception as e:  
 ¦ print(f'simplifier failure: {e}')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 #################################  
 导出pytorch为onnx模型，并命名为"od"  
 #################################  
 simplifyONNX(od)

***onnxsim不仅可以去除onnx模型冗余操作（减小模型大小）和进行一定优化，某些时候转MNN必须要先进行onnxsim优化***