**Tensor.detach()**

作用是阻断反向梯度传播，在训练网络的时候保持一部分的网络参数不变，不让其梯度对主网络的梯度造成影响

返回一个新的 tensor，其requires\_grad为False，且与原计算图形分离。新的张量与原始张量共享相同的存储，若对二者中某一个中的元素进行修改，另一个中的对应元素也会发生改变  
a = torch.tensor([1.0, 2.0, 3.0],requires\_grad=True)

print(a.grad)

out1 = a.sum()

out1.backward()

print(a.grad)

a.grad.data.zero\_()

>>>

None

tensor([1., 1., 1.])

接上面代码

c = out1.detach()*#添加detach()，c的requires\_grad为False*

p = torch.tensor([1.0,2.0,3.0],requires\_grad=True)

out2 = (c+p).sum()

out2.backward()

print(p.grad)

print(a.grad)

>>>

tensor([1., 1., 1.])

tensor([0., 0., 0.])