torch.save()有三种保存方式：

1.只保存神经网络的训练模型的参数，save的对象是model.state\_dict()，其类型是OrderDict

2.既保存整个神经网络的的模型结构又保存模型参数，save的对象就是整个模型

3.通过保存模型的完整状态作为checkpoint，在模型训练失败时，可以从保存的检查点模型继续训练，以避免训练失败时每次都需要从头开始带来的训练时间浪费

PyTorch模型保存的文件扩展名可以是.pt/.pth/.pkl

**只保存模型参数**

model = nn.Linear(10,1)*#训练好的网络*

torch.save(model.state\_dict(), 'net\_params.pth')*#只保存模型参数*

model = nn.Linear(10,1)*#在另一文件中先声明模型结构*

model.load\_state\_dict(torch.load('net\_params.pth'))*#仅加载参数*

**保存整个模型**

model = nn.Linear(10,1)

torch.save(model, 'net.pth')*#保存整个神经网络的模型结构以及参数*

model = torch.load('net.pth')*#在另一文件中直接加载整个神经网络的模型结构以及参数*

**保存checkpoint：**

*#在保存checkpoint中，可以保存模型参数，也可以保存整个模型，此处展示保存模型参数的范例*

def save\_checkpoint (state,file\_name):

print('saving check\_point')

torch.save(state,file\_name)

def load\_checkpoint(checkpoint):

print('Load \_model')

model.load\_state\_dict(checkpoint['state\_dict'])

optimizer.load\_state\_dict(checkpoint['optimizer'])

*#然后我们需要在训练模型的过程中把checkpoint保存下来*

for epoch in num\_epoch:

*#设置每隔多少个epoch保存一次*

if epoch% 2 == 0:

checkpoint = {'state\_dict': model.state\_dict(), 'optimizer': optimizer.state\_dict()}

save\_checkpoint(checkpoint,'cnn\_model.pth')

*#读取的时候要在训练过程的前面，模型和优化器定义的后面*

pre\_train = True

if pre\_train:

load\_checkpoint(torch.load('cnn\_model.pth'))